EXPLORING STRUCTURE AND DYNAMICS OF COMPLEX NETWORKS:
NOVEL METHODS AND INTERDISCIPLINARY APPLICATIONS

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Yuriy Hulovatyy

Tijana Milenković, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
June 2016
EXPLORING STRUCTURE AND DYNAMICS OF COMPLEX NETWORKS: NOVEL METHODS AND INTERDISCIPLINARY APPLICATIONS

Abstract

by

Yuriy Hulovatyy

Networks provide a natural and powerful way to model complex real-world systems in various domains. Studying structure of a network can help extract functional knowledge about the corresponding system. As real-world networks exhibit non-trivial organization at many scales, this extraction can be done on different levels: from the global perspective of the whole network to the intermediate perspective of node groups (or communities) to the local perspective of individual nodes. With new technological advances, the amount of available real-world network data in different domains rapidly increases. In addition, networks are growing in size and complexity. For example, whereas traditional network data has been static, because it has become easier to record system evolution, more of dynamic network data is becoming available. For these reasons, it is critical to develop novel computational strategies for efficient extraction of functional information from the structure of such complex (e.g., dynamic) networks. And this is the main focus of this dissertation.

We achieve this goal in two different ways, by: 1) answering novel research questions via established network approaches, and 2) developing novel network approaches for established research questions.

In the first context, we apply global network analysis to answer a novel question in a novel domain in which network research has not been used to date – interpreting
affective physiological data. In addition, we employ local network analysis to study
the interplay between individuals’ social interactions and traits from a new dynamic
(rather than traditional static) network viewpoint.

In the second context, we take a well-established local analysis approach for static
networks to develop a novel method for the problem of link prediction, which we use
for de-noising biological networks. Moreover, we take the same static local approach
and develop new theory for dynamic network analysis. We demonstrate that ac-
counting for temporal information helps and use our method to study human aging
from biological networks. Finally, we introduce a new approach for studying dynamic
networks from the intermediate perspective, which deals with the problem of segment
community detection. We show that our approach outperforms existing methods in
terms of both accuracy and computational complexity.
To my family
CONTENTS

FIGURES .................................................................................. vii

TABLES .................................................................................. xii

ACKNOWLEDGMENTS ............................................................... xiv

CHAPTER 1: INTRODUCTION ....................................................... 1

I NOVEL RESEARCH QUESTIONS VIA ESTABLISHED NETWORK AP-
PROACHES .............................................................................. 7

CHAPTER 2: IMPROVING INTERPRETATION OF AFFECTIVE PHYSI-
OLOGICAL DATA VIA GLOBAL NETWORK ANALYSIS ................. 8

2.1 Introduction ........................................................................... 8

2.1.1 Motivation and Background ............................................ 9

2.1.2 Related Work .................................................................. 9

2.1.3 Our Approach and Contributions .................................... 10

2.2 Methods ............................................................................... 11

2.2.1 Data Collection .............................................................. 13

2.2.2 Network Construction .................................................... 15

2.2.3 Network Properties ........................................................ 19

2.2.4 Clustering of Networks ................................................... 19

2.2.5 Evaluating Partition Quality .......................................... 23

2.3 Results and Discussion ....................................................... 25

2.3.1 Results for Unfiltered Networks .................................... 25

2.3.2 Results for Filtered Networks ........................................ 33

2.3.3 Comparison of Results for Unfiltered and Filtered Networks 33

2.3.4 In-depth Analysis and Interpretation of a Representative Network-
based Image Partition ......................................................... 39

2.4 Conclusions ......................................................................... 43

2.5 Computational Novelty of the Work ..................................... 44

2.6 My Contributions to the Work ............................................. 44
CHAPTER 3: STUDYING THE INTERPLAY BETWEEN INDIVIDUALS’ EVOLVING INTERACTION PATTERNS AND TRAITS IN DYNAMIC SOCIAL NETWORKS VIA LOCAL NETWORK ANALYSIS

3.1 Introduction
3.1.1 Motivation
3.1.2 Related Work
3.1.3 Our Approach and Contributions

3.2 Methods
3.2.1 Dataset
3.2.2 Network Construction
3.2.3 Measures of Network Topology
3.2.4 Network-based Evolution Clustering
3.2.5 Evaluating Partition Quality

3.3 Results and Discussion
3.3.1 Detection of Network-Trait Interplay
3.3.2 Validation of Our Results
3.3.3 Robustness of Our Approach
3.3.4 Comparison of Our Results with Findings from the Literature
3.3.5 On Dynamic versus Static Network Analysis

3.4 Conclusions
3.5 Computational Novelty of the Work
3.6 My Contributions to the Work

II NOVEL NETWORK APPROACHES FOR ESTABLISHED RESEARCH QUESTIONS

CHAPTER 4: REVEALING MISSING PARTS OF THE CELLULAR INTERACTOME VIA GRAPHLET-BASED LINK PREDICTION

4.1 Introduction
4.1.1 Motivation
4.1.2 Related Work
4.1.3 Our Approach and Contributions

4.2 Methods
4.2.1 Network Data
4.2.2 Existing Commonly Used LP Measures
4.2.3 New LP Measures
4.2.4 Evaluation Framework

4.3 Results and Discussion
4.3.1 Test 1: Evaluating LP Methods by Introducing Synthetic Noise into PPI Networks
4.3.2 Test 2: Evaluating LP Methods on HC Network with respect to Low-confidence PPI Data
4.3.3 Test 3: De-noising the PPI Networks
APPENDIX A: FOR CHAPTER 2
  A.1 IAPS Identifiers

APPENDIX B: FOR CHAPTER 3
  B.1 Appendix Figures
  B.2 Appendix Tables

APPENDIX C: FOR CHAPTER 4
  C.1 Density-weighted Node-pair-GDV-centrality
  C.2 Evaluation Framework
  C.3 Appendix Figures
  C.4 Appendix Tables

APPENDIX D: FOR CHAPTER 5
  D.1 Computing the Number of Dynamic Graphlet Types of a Given Size
  D.2 Constrained Counting of Dynamic Graphlets in a Network
  D.3 Experimental Setup
  D.4 Results: the Effect of Graphlet Size
  D.5 Appendix Figures
  D.6 Appendix Tables

APPENDIX E: FOR CHAPTER 6
  E.1 Related Work
  E.2 Our SCOUT Approach
    E.2.1 Objective Function
    E.2.2 Consensus Clustering
    E.2.3 Search Strategy
  E.3 Experimental Setup
    E.3.1 Methods for Comparison
    E.3.2 Datasets
    E.3.3 Evaluation Measures
    E.3.4 Statistical Significance of Two Methods’ Performance Difference
  E.4 Results: the Effect of Method Parameter Choices
  E.5 Appendix Figures
  E.6 Appendix Tables

BIBLIOGRAPHY
FIGURES

2.1 Overview of our 4-step study ........................................ 12
2.2 Illustration of physiological data used in our study ............ 12
2.3 Edge intersections .................................................. 26
2.4 Illustration of topological properties of unfiltered networks .... 28
2.5 Illustration of differences in topological properties of unfiltered and filtered networks ........................................... 35
2.6 Arousal-valence scores of different clusters in the representative network-based partition ........................................ 42
2.7 Membership of the 89 images in different clusters of the representative network-based partition .......................... 42

3.1 Classification of the related work based on the three dimensions . 47
3.2 Summary of our work .............................................. 50
3.3 Statistics of a network of the given type at the given timeslot for the NetSense dataset .................................................... 53
3.4 Interplays between a network type and a trait detected in our NetSense dataset via at least one centrality by AID-ED ............ 59
3.5 Statistically significantly correlated trait pairs according to at least one of AID and WRST on the NetSense dataset ............. 61
3.6 Interplays between a network type and a trait detected in our NetSense dataset via at least one centrality .......................... 63
3.7 Interplays between a network type and a trait detected in our NetSense dataset and the friends and family dataset ............. 67

4.1 Graphlet positions of a node, an edge, a non-edge, and a node pair . 84
4.2 LP accuracy of the graphlet-based methods in the context of evaluation test 1 ......................................................... 93
4.3 Comparison of different methods in the context of evaluation test 1 . 96
4.4 Comparison of different methods in the context of evaluation test 2 . 98
4.5 Comparison of different methods in the context of evaluation test 3 . 100

5.1 Illustration of the difference between static and dynamic graphlets . 112
5.2 Illustration of how we extend a dynamic graphlet with an additional more recent event ............................................... 114
5.3 Illustration of our dynamic graphlet counting procedure ............ 116
5.4 Comparison of the graphlet approaches in the context of network and node classification ........................................... 122
6.1 Illustration of SCD .......................................................... 135
6.2 Illustration of how SCD naturally allows for compromising between
the two extremes of snapshot clustering and consensus clustering . . 136
6.3 The process of constructing a node-time partition $\mathcal{P}_B$ when computing $\text{Sim}_B$ ................................................................. 148
6.4 Representative $\text{Sim}_T$ and $\text{Sim}_P$ scores for synthetic networks with 100
nodes per snapshot and four ground truth segments .................... 155
6.5 Representative method comparison for synthetic networks with 100
nodes per snapshot ................................................................. 156
6.6 Rankings of the methods for synthetic networks with respect to $\text{Sim}_B$ 159
6.7 Running times of the methods for synthetic networks ............... 161
6.8 Method comparison for real-world networks .............................. 163
6.9 The top 10 highest ranked time points by each method for Senate
network ................................................................. 166
6.10 Representative $Q_P$ scores of SCOUT’s solutions for different numbers
of segments $l$ and the solutions of the existing methods for High School
network and Senate network ............................................. 168
6.11 Running times of the methods for real-world networks ................ 169

B.1 Statistics of a network of the given type at the given timeslot for the
friends and family dataset .................................................... 179
B.2 Statistically significantly correlated trait pairs according to at least
one of AID and WRST for the friends and family dataset ....... 180
B.3 Interplays between a network type and a trait for the NetSense dataset
and the equal size trait-based partitioning strategy detected via at least
one centrality by AID-ED ..................................................... 180
B.4 Interplays between a network type and a trait for the friends and family
dataset and the $k$-medoids trait-based partitioning strategy detected
via at least one centrality by AID-ED ..................................... 181
B.5 Interplays between a network type and a trait for the friends and family
dataset and the equal size trait-based partitioning strategy detected
via at least one centrality by AID-ED ..................................... 181

C.1 Link prediction accuracy of the unweighted graphlet-based methods in
terms of AUROCs at different noise levels as $\alpha$ is varied from 0 to 1 . 188
C.2 Link prediction accuracy of the density-weighted graphlet-based meth-
ods in terms of AUROCs at different noise levels as $\alpha$ is varied from 0
to 1 ................................................................. 189
C.3 Link prediction accuracy of the graphlet-based methods in terms of
AUROCs at different noise levels comparing unweighted and weighted
versions of the methods ......................................................... 190
C.4 Link prediction accuracy of the weighted graphlet-based methods in
terms of AUROCs at different noise levels comparing different graphlet
sizes in each of the three networks (AP/MS, HC, and Y2H) .......... 191
C.5 Link prediction accuracy of our new methods and existing methods in terms of AUROCs at different noise levels comparing the different methods in each of the three networks (AP/MS, HC, and Y2H).

C.6 Receiver operator curves for the AP/MS network for each noise level.

C.7 Receiver operator curves for the HC network for each noise level.

C.8 Receiver operator curves for the Y2H network for each noise level.

C.9 Link prediction accuracy of the unweighted graphlet-based methods in terms of AUPRs at different noise levels as $\alpha$ is varied from 0 to 1.

C.10 Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUPRs at different noise levels as $\alpha$ is varied from 0 to 1.

C.11 Link prediction accuracy of the unweighted graphlet-based methods in terms of AUPRs at different noise levels comparing unweighted and weighted versions of the methods.

C.12 Link prediction accuracy of the weighted graphlet-based methods in terms of AUPRs at different noise levels comparing different graphlet sizes in each of the three networks (AP/MS, HC, and Y2H).

C.13 Link prediction accuracy of our new methods and existing methods in terms of AUPRs at different noise levels comparing the different methods in each of the three networks (AP/MS, HC, and Y2H).

C.14 Precision-recall curves for the AP/MS network for each noise level.

C.15 Precision-recall curves for the HC network for each noise level.

C.16 Precision-recall curves for the Y2H network for each noise level.

C.17 F-score curves for the AP/MS network for each noise level.

C.18 F-score curves for the HC network for each noise level.

C.19 F-score curves for the Y2H network for each noise level.

C.20 Link prediction accuracy of the unweighted graphlet-based methods in terms of AUROCs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1.

C.21 Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUROCs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1.

C.22 Link prediction accuracy of the graphlet-based methods in terms of AUROCs for the HC network against the low confidence data comparing unweighted and weighted versions of the methods.

C.23 Link prediction accuracy of the graphlet-based methods in terms of AUROCs for the HC network against the low confidence data comparing different graphlet sizes.

C.24 Receiver operator curves for the HC network against the low confidence data comparing the different methods.

C.25 Link prediction accuracy of the unweighted graphlet-based methods in terms of AUPRs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1.
C.26 Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUPRs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 210
C.27 Link prediction accuracy of the graphlet-based methods in terms of AUPRs for the HC network against the low confidence data comparing unweighted and weighted versions of the methods 211
C.28 Link prediction accuracy of the graphlet-based methods in terms of AUPRs for the HC network against the low confidence data comparing different graphlet sizes 212
C.29 Precision-recall curves for the HC network against the low confidence data comparing the different methods 212
C.30 Pairwise intersections between each pair of de-noised networks as well as between each de-noised network and the original network, for each of the three networks (AP/MS, HC, and Y2H) 213
C.31 Percentage of new predicted edges from AP/MS that are validated in BioGRID 214
D.1 Comparison of the different graphlet approaches in the context of biological network classification 229
D.2 Pairwise similarities between the different methods and their parameter variations in the test of node classification 230
D.3 Comparison of the different graphlet approaches in the context of aging-related node classification when considering DyNetAge “ground truth” aging-related data 231
D.4 Comparison of the different graphlet approaches in the context of aging-related node classification when considering BrainExpression2004Age, a non-network “ground truth” aging-related data obtained via gene expression data analyses 231
D.5 Comparison of the different graphlet approaches in the context of aging-related node classification when considering SequenceAge, a non-network “ground truth” aging-related data obtained via genomic sequence analyses 232
D.6 Overlap between aging-related predictions produced by the four graphlet approaches at the first value of $k$ and the second value of $k$ 233
E.1 The procedure of obtaining solution $O_{i,l}^*$ for $D_{[0,i-1]}$ that has $l$ segments during the $i^{th}$ iteration of the exhaustive search 269
E.2 The top-down search 270
E.3 The bottom-up search 270
E.4 The process of constructing synthetic dynamic network $D$ with known ground truth solution $O^{(gt)} = (T^{(gt)}, P^{(gt)})$ 271
E.5 The effect of $\theta$ value for Multi-Step 272
E.6 The effect of $w$ value for GHRG 272
E.7 The effect of $Q_B$ choice for SCOUT when choosing the optimal number of segments 273
E.8 The effect of $Q$ choice for SCOUT when producing the best solution for each possible number of segments .
E.9 The effect of the choice of static clustering method in the sum graph consensus clustering method for SCOUT.
E.10 The effect of the choice of consensus clustering method for SCOUT.
E.11 The effect of the choice of search strategy for SCOUT.
E.12 Rankings of the methods with respect to $Sim_T$ and change point classification.
E.13 $Sim_T$ and $Sim_P$ scores for all 20 synthetic network configurations.
E.14 The number of segments $l$ in the solutions produced by the methods for synthetic networks.
E.15 Method comparison for synthetic networks with 50, 500, and 1000 nodes per snapshot.
E.16 Rankings of the methods with respect to $Q_P$ and $Sim_P$.
E.17 The number of segments $l$ in the solutions produced by the methods for real-world networks.
E.18 $Q_P$ scores of SCOUT’s solutions for different numbers of segments $l$ and the solutions of the existing methods.
TABLES

2.1 NETWORK TYPES THAT RESULT IN "STATISTICALLY SIGNIFICANT" IMAGE PARTITIONS .................................................... 30
2.2 QUALITY OF THE "STATISTICALLY SIGNIFICANT" IMAGE PARTITIONS RESULTING FROM CLUSTERING OF UNFILTERED NETWORKS .......................................................... 31
2.3 QUALITY OF THE "STATISTICALLY SIGNIFICANT" IMAGE PARTITIONS RESULTING FROM CLUSTERING OF FILTERED NETWORKS ........................................................................ 32
3.1 BIG FIVE PERSONALITY TRAIT DESCRIPTIONS .................. 53
3.2 INTERPLAYS BETWEEN A NETWORK TYPE AND A TRAIT DETECTED WITH SEVEN CENTRALITIES ON THE NetSense DATASET ................................................................. 57
3.3 VALIDATION RATES OF OUR RESULTS ON THE NetSense DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITIES ................................................................. 65
5.1 PRECISION OF THE DIFFERENT METHODS IN THE CONTEXT OF AGING ................................................................. 127
5.2 OVERLAPS OF GENES, ENRICHED FUNCTIONS, AND ENRICHED DISEASES ................................................................. 128
6.1 STATISTICAL SIGNIFICANCE OF THE SUPERIORITY OF SCOUT OVER THE BEST EXISTING METHOD IN TERMS OF SimB ON SYNTHETIC NETWORKS ................................................................. 160
B.1 VALIDATION RATES OF OUR RESULTS ON THE NetSense DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITY ................................................................. 182
B.2 VALIDATION RATES OF OUR RESULTS ON THE NetSense DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITY, WHEN TAKING INTO ACCOUNT ONLY THE BIG FIVE PERSONALITY TRAITS ONLY ................................................................. 183
B.3 VALIDATION RATES OF OUR RESULTS ON THE friends and family DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITY, WHEN TAKING INTO ACCOUNT ONLY THE BIG FIVE PERSONALITY TRAITS ONLY ................................................................. 184
C.1 A SAMPLE OF THE LIST OF ALL NODE PAIRS PREDICTED AS EDGES BY ANY METHOD .................................................. 215

D.1 THE NUMBER OF DYNAMIC GRAPHLETS WITH A GIVEN NUMBER OF EVENTS THAT HAVE AS THEIR BACKBONE THE SAME STATIC GRAPHLET, FOR ALL STATIC GRAPHLETS WITH UP TO FIVE NODES ................................................................. 234

D.2 THE NUMBER OF DYNAMIC GRAPHLET TYPES WITH UP TO TEN NODES AND TEN EVENTS ............................................ 236

D.3 DETAILED NETWORK CLASSIFICATION RESULTS FOR THE DIFFERENT METHODS AND DIFFERENT PARAMETERS IN EACH METHOD .................................................................................................................................................. 237

D.4 RUNNING TIMES OF THE GRAPHLET METHODS FOR EACH OF THE NINE NETWORK CLASSES IN THE NETWORK CLASSIFICATION TASK .................................................................................................................................................. 238

D.5 DETAILED NODE CLASSIFICATION RESULTS FOR THE DIFFERENT METHODS AND DIFFERENT PARAMETERS IN EACH METHOD .................................................................................................................................................. 239

D.6 PRECISION OF THE DIFFERENT METHODS IN THE CONTEXT OF AGING AT THE TWO $k$ VALUES .................................................. 240

D.7 A SAMPLE OF THE LIST OF AGING-RELATED PREDICTIONS PRODUCED BY THE FOUR GRAPHLET APPROACHES AT THE TWO VALUES OF $k$ .................................................................................................................. 241

D.8 GO TERMS ENRICHED IN NOVEL AGING-RELATED PREDICTIONS OF DYNAMIC GRAPHLETS AT THE SECOND VALUE OF $k$ .................................................................................................................. 242

E.1 REAL-WORLD NETWORKS THAT WE USE IN OUR STUDY ........ 284
ACKNOWLEDGMENTS

This dissertation is about networks and also would not have been possible without one. During my Ph.D. studies, I have been fortunate to receive help and guidance from so many people around me, from my social network. It is thus a great pleasure to express my sincere gratitude to everyone who supported me along this journey.

First and foremost, I would like to thank my advisor Prof. Tijana Milenković for her guidance, patience, and support. I am grateful to her for showing me how to formulate and approach a research problem, providing me the freedom of carrying out research at all stages of the research cycle, and all the time she spent helping me improve my writing and presentation skills. Her advice, expertise, and outstanding work ethic have been invaluable to my research and professional development.

I would also like to thank Prof. Nitesh Chawla, Prof. Sidney D’Mello, and Prof. Aaron Striegel for generously taking the time to serve on my dissertation committee, providing valuable feedback, and asking insightful questions.

In addition to this, I would like to thank all my friends and colleagues at the University of Notre Dame who made this journey more enjoyable. I am particularly thankful to the members of CoNe Lab for interesting discussions and useful feedback.

Last but not least, I am grateful to my family for their unlimited and unconditional love and support throughout my life. Special thanks go to my parents, grandparents, sister Julia, and brother-in-law Russell for their endless encouragement and belief in me. The whole family has been a constant source of inspiration for me, and without them, I would never be who I am now.
CHAPTER 1

INTRODUCTION

Many complex systems, from biological cells to societies to the Internet, consist of multiple elements and some relationships between the individual elements. Hence, networks (or graphs), which model the elements as nodes and their relationships as edges, provide an elegant yet powerful way to reason about the real-world systems. This reasoning is typically done by extracting functional knowledge about the systems from their network structure (or topology). This dissertation focuses on developing novel efficient computational strategies for this purpose.

As real-world networks exhibit structure at many scales, extracting functional information from network topology is typically done on three different levels: global (or macroscopic), local (or microscopic), or intermediate (or mesoscopic). Each of these levels of studying network structure has its (dis)advantages, depending on the research question of interest, as follows.

The goal of macroscopic network analysis is to measure and describe the overall structure of a network via global measures such as network density, diameter, clustering coefficient, or degree distribution [145]. In this context, we use state-of-the-art global network approaches to answer a novel question in a novel domain in which network research has not been used to date – interpreting affective physiological data – in hope that we would do so better than existing statistical approaches that have been used for this purpose thus far. Studying this question is important because understanding how human physiological (e.g., galvanic skin) responses to stimuli (e.g., emotionally charged images) vary across individuals is critical for the field of affective
computing [160].

While the global measures can summarize the structure of the entire network as a whole, they are not mathematically sensitive enough to capture detailed topological characteristics of complex real-world networks. For example, two networks with very different structures might have the same value of a global network property and could thus not be distinguished based on that property [131]. For these reasons, network research has shifted towards microscopic network analysis via local properties such as motifs (statistically significantly overrepresented partial subgraphs) [136], graphlets (small induced subgraphs) [165], or node centralities [145]. Local properties have been useful in linking topological position of an individual node or a very small (up to 5-node) network region to its function, via notions of, e.g., network alignment or clustering [127, 130, 132, 133]. In this context of local network analysis, we use the well-established concept of graphlets to develop a novel method for a popular computational problem – that of link prediction. This problem has many real-world applications, including de-noising of biological (cellular) networks by predicting missing and spurious links, which is the main motivation of our work. As many existing methods for link prediction rely on limited topological information (e.g., shared immediate neighborhoods of the nodes to be linked), they have drawbacks [114]. Therefore, we study in more detail what are the topological properties of nodes that dictate whether the nodes should be linked and introduce novel sensitive and as we show superior methods with the goal of capturing these properties. In the process, we demonstrate that link prediction can improve biological correctness of cellular networks, which is its ultimate goal in the field of computational biology.

Existing network research, both global and local, including the graphlet methodologies, has traditionally dealt with static network representations of real-world systems of interest, owing mostly to drawbacks of technologies for data collection that until only recently have been limited to gathering static data only. While this has
led to considerable advances in understanding of the real-world systems [145], such a static view overlooks an important aspect of many real-world systems – the fact that they evolve over time. For example, environment of a biological cell is dynamically changing, people are interacting with different people at different times, and information is spreading over the Internet. With recent technological advances, more and more of dynamic (or temporal) real-world network data is becoming available. The question is how to study the resulting dynamic data efficiently, given that most of the traditional network research has focused on developing methods for analyzing static and not dynamic network data. An obvious approach is to aggregate the dynamic data into a single network snapshot and then apply the existing approaches to this static representation. However, doing so would lead to loss of important temporal information about evolution of network structure [77]. Hence, there is a critical need to develop new approaches for dynamic network analysis that would explicitly account for the temporal information. And this is exactly what we aim to address next in this dissertation, as follows.

First, we employ a “static-temporal” approach, which: splits a network into a series of snapshots, locally analyzes each snapshot, and tracks snapshot-based changes in local network structure over time. We use this approach in the context of studying how evolving network positions of individuals in a dynamic social network (encompassing, e.g., face-to-face proximities, phone calls, and Facebook postings) relate to the individuals’ physical and personal traits (such as weight, health, and happiness). Studying this is important to answer to what extent does one shape the social network(s) one participates in, or to what extent do the social networks shape the individual [126] [192]. Even though the interplay between individuals’ social interactions and traits has been studied extensively, traditionally this was done from static social network data. We show that our static-temporal framework detects large amounts of interplay that cannot be captured by a static network approach alone or
by traditional non-network approaches that have been used for this purpose.

Nonetheless, while the static-temporal approach allows for studying evolution of network structure better than the aggregated approach, it still treats each snapshot in isolation, thus ignoring important temporal information about the relationships between different snapshots. With this motivation, i.e., with the goal of capturing these relationships, as our second contribution within the topic of dynamic network analysis, we take the established notion of static network graphlets to the next level to develop new graphlet-based theory that allows for dynamic network analysis. Our results on synthetic and real-world social as well as biological networks demonstrate that the resulting notion of dynamic graphlets is superior compared to static and even static-temporal graphlet-based approaches, in the sense that our novel dynamic graphlet approach distinguishes better between different network types or functionally different nodes than the other two approaches. We apply our dynamic graphlet approach to biological networks to study cellular changes with age and consequently to computationally predict novel knowledge about human aging, which is hard to study experimentally due to long lifespan and ethical constraints.

Thus far, we have discussed the study of network structure from either global or local perspective. Whereas both perspectives have received significant attention, there is also rich information at the intermediate network structural level – that of communities (or clusters). While there is no single universally accepted formal definition, intuitively, community is a group of topologically or functionally related nodes in a network. Thus, efficient identification and analysis of network communities can help describe network structure at a node group level or identify functional units in the system that is modeled by the network. For example, clusters can correspond to proteins performing the same function in the cell, people with shared interests, or web pages dealing with related topics. In the static setting, community detection has been an active area of research with a zoo of methods proposed for this purpose.
With the increasing availability of temporal network data, initial efforts for detecting and analyzing dynamic communities in such networks have been made [72]. While the problem of community detection is not universally defined even in the “simple” static setting, temporal dimension in the dynamic data introduces another level of complexity. Typically, community detection in the dynamic setting is approached by assuming either that each time point has a distinct community organization or that all time points share one community organization. In reality, the truth likely lies between these two extremes, since some time periods can have community organization that evolves while others can have community organization that stays the same. To find the compromise, we consider dynamic community detection in the context of the problem of segment detection, which identifies contiguous time periods with consistent network structure [172]. Consequently, we formulate a combined problem of segment community detection (SCD), which simultaneously partitions the network into contiguous time segments with consistent community organization and finds this community organization for each segment. To solve SCD, we introduce SCOUT, an optimization framework that explicitly considers both segmentation quality and partition quality. SCOUT addresses limitations of existing methods that can be adapted to solve SCD, which typically consider only one of segmentation quality or partition quality. We demonstrate that SCOUT outperforms the existing methods in terms of accuracy, while also being more computationally efficient.

In summary, this dissertation explores the following research directions:

1. Novel research questions via established network approaches (Part I):
   (a) Improving interpretation of affective physiological data via global network analysis (Chapter 2).
   (b) Studying the interplay between individuals’ evolving interaction patterns and traits in dynamic social networks via local network analysis (Chapter 3).

2. Novel network approaches for established research questions (Part II):
   (a) Revealing missing parts of the cellular interactome via graphlet-based link
prediction (Chapter 4).

(b) Exploring the structure and function of temporal networks with dynamic graphlets (Chapter 5).

(c) SCOUT: simultaneous time segmentation and community detection in dynamic networks (Chapter 6).
PART I

NOVEL RESEARCH QUESTIONS VIA ESTABLISHED NETWORK APPROACHES
CHAPTER 2

IMPROVING INTERPRETATION OF AFFECTIVE PHYSIOLOGICAL DATA
VIA GLOBAL NETWORK ANALYSIS

The work presented in this chapter has been published as follows:


2.1 Introduction

Networks (or graphs) allow for studying complex processes that emerge from the collective behavior of interconnected elements. Thus, networks can model real phenomena in many domains, e.g., social, technological, or biological systems [128, 130, 155, 215]. We focus on network modeling of affective (or emotional) physiological data in order to gain insights into how individuals physiologically respond to emotional stimuli, thereby benefiting the fields of Affective Psychophysiology and Affective Computing. While networks have already been used to evaluate emotional responses (e.g., [141]), to our knowledge, we are the first to apply them to study affective physiological data. As we will show, network analysis improves interpretation of the data compared to an alternative non-network-based approach.
2.1.1 Motivation and Background

There is an inextricable coupling between physiology and emotions because one of the key evolutionary functions of emotion is to facilitate rapid action in response to relevant environmental events [87]. Emotions are constructs (or conceptual entities) that cannot be directly measured, but must be inferred from measurable signals like physiology. Thus, understanding the link between emotions and physiology has been an important endeavor in the field of Affective Psychophysiology for more than a century.

There is also an engineering side to complement the scientific endeavor of identifying the physiological correlates of affect. The field of Affective Computing, a subfield of Human-Computer Interaction, aims to build intelligent systems that respond to user emotions much like an actual human would [160]. For example, a system can offer a hint if it detects that a user is confused or frustrated. Considerable work has focused on developing automated approaches to detect emotions from observable signals like facial expressions, speech patterns, etc. (e.g., [27, 227]). Physiological-based approaches for affect detection are attractive (e.g., [5, 29, 161]), as these signals are largely involuntary and thereby less susceptible to social masking like facial expressions and speech. This once again raises the fundamental issue of understanding the relationship between affect and physiology, which is the focus of this study.

2.1.2 Related Work

Over the last century, many attempts have been made to identify how different emotions are manifested in physiological signals, such as the electrocardiogram (ECG), electromyogram (EMG), or galvanic skin response (GSR) (see [109] for a review). While it was once claimed that unique discrete emotions (e.g., fear, anger) are accompanied by distinct physiological patterns [51], meta-analyses and other syntheses of the literature have failed to conclusively support this claim [109].
One reason for the difficulties in uncovering the emotion → physiological mapping is the considerable individual variability in emotional responses (i.e., reactions to the same stimulus vary across individuals). In affect detection, the most common approach to handle this variability has been to simply ignore it (e.g., \[29, 74, 117\]). This is done by building person-dependent models that are carefully calibrated to each individual. However, despite some advances \[161, 207\], what is needed are person-independent models that generalize to new individuals. The few efforts along this front have produced mixed results \[95\]. For example, emotion recognition accuracy of person-dependent and person-independent models from three physiological signals (ECG, EMG, and GSR) were compared \[5\], and machine learning applied to detect seven emotions indicated that it was possible to build person-dependent models with moderate classification accuracy, but accuracy went to zero with person-independent modeling.

2.1.3 Our Approach and Contributions

To summarize, efforts to identify unique emotion-specific physiological responses have been largely unsuccessful, likely due to considerable intra- and inter-individual variability in the signals. Unfortunately, most (but not all) of the research have typically considered this variability to be sources of error and something to be averaged over. In our view, however, this variability is far from random as there might be structure in the noisy patterns of individual responding. Modeling this variability will provide insights into the fundamental question of how individuals physiologically respond to emotions.

This work adopts a novel approach to study interrelations among individuals by formulating the problem from a network perspective. Since networks model relationships between objects, and since physiological variability depends on the inter-relations – both between humans and their physiological states – by looking at the
system of stimuli and individual responses to them from a structural (or topological) point of view, networks can provide important insights on the problem of modeling and understanding this variability.

To this end, we consider affective physiological responses of humans to a set of stimuli and construct networks reflecting relations between individual responses. We then analyze and compare these networks to investigate how inter-individual patterns of responses map onto the known “ground truth” about the stimuli. In addition to this, we provide a comprehensive analysis of the ways to construct, compare, and cluster networks, which allows us to examine relative effects of choosing different methods and parameters. As a result, we develop an extensible framework for systematic analysis of affective physiological data. Moreover, we demonstrate that network analysis of such data improves interpretation of the data compared to an alternative, non-network-based approach. Finally, we analyze in detail a representative network-based image partition and discuss practical applications of its results.

2.2 Methods

Our study consists of four major steps (Figure 2.1). First, we obtain affective physiological data (Section 2.2.1). Second, we construct networks from this data using a series of network inference strategies (Section 2.2.2). Third, we use various clustering methods and combinations of their parameters to partition the set of networks based on their topological similarities (Sections 2.2.3 and 2.2.4). Fourth, we conduct a thorough evaluation of the partitions produced by the different clustering strategies, and verify both statistical and practical significance of our results (Section 2.2.5).
Figure 2.1: Overview of our 4-step study. Each of the four steps is discussed in one of the four subsections of Section 2.2.

Figure 2.2: Illustration of physiological data used in our study with respect to: (a) arousal/valence (AV) “ground truth” partition of the image set and (b) experimental setup. In panel (a), for each image, its IAPS arousal normative score was categorized as either “low”, “medium”, or “high”, and its IAPS valence normative score was categorized as either “negative”, “neutral”, or “positive”. Based on this $3 \times 3$ arousal/valence space, images were partitioned into nine classes.
2.2.1 Data Collection

During the experiment, 18 human subjects were presented with 89 emotionally charged images from the International Affective Picture System (IAPS) \[39, 40, 108\]. The IAPS is a collection of over a 1,000 images depicting people, objects, or events that have been selected on the basis of how they evoke valence (unpleasant to pleasant) and arousal (sleepy to active) in large samples of viewers. Arousal and valence are the two fundamental dimensions of affective responses \[178\].

The 89 images were selected to cover a $3 \times 3$ arousal-valence (arousal: low, medium, high; valence: positive, neutral, negative) normative rating space with 10 images per each of the nine classes (one class had only nine images due to error) (Figure 2.2(a)). For example, the images in the positive valence and high arousal class were selected using the normative ratings above 6.0 for valence and above 5.5 for arousal (normative ratings ranged from 1 to 9). Selection of images proceeded in a three-step process. In Step 1, the images were categorized into the nine classes of the $3 \times 3$ valence-arousal space based on the normative IAPS ratings. In Step 2, the researchers then manually selected images from each class to ensure that there was sufficient variability in content and minimal overlap across classes. In Step 3, mean valence and arousal scores for the selected images in each class were analyzed in order to ensure that they were sufficiently different. For reproducibility of our study, IAPS identifiers of the selected 89 images, along with their corresponding classes in the arousal-valence partition, are presented in Appendix Section A.1.

Henceforth, we refer to this nine-class image partition as the arousal/valence (AV) “ground truth” partition (or simply as the AV partition). Note that we intentionally use quotes when talking about the AV partition, since the “ground truth” is itself quite noisy. This is because the “ground truth” was obtained from a combination of theory (images were carefully selected to evoke particular responses; e.g., spider to evoke fear, or surgery for disgust) and normative ratings of valence and arousal.
that accompany the IAPS collection. These ratings reflect the average valence and arousal as subjectively reported by a large sample of individuals (different from our 18 subjects) after viewing each image. Therefore, the assignment of the images to the nine “ground truth” classes in the AV partition is noisy, which is typical for any emotion-eliciting stimulus.

For physiological recording, participants were equipped with electrocardiogram (ECG), electromyogram (EMG), respiration, and galvanic skin response (GSR) sensors (Figure 2.2(b)) [142]. The BIOPAC MP1501 system with AcqKnowledge software was used to acquire the physiological signals at a sampling rate of 1,000 Hz for all channels. Two electrodes were placed on the wrists for collecting ECG. Two channels recorded EMG activity from two facial muscles regions (zygomatic and corrugator, respectively). GSR was recorded from the index and middle finger of the left hand, and a respiration band was strapped around the chest.

We exclusively focus on the GSR signal as an initial step to studying the viability and added-value of the network-based approach to analyzing affective psychophysiological data. The GSR was used in lieu of the other signals due to the considerable research establishing a clear link between GSR and physiological arousal; this link is less clear for the other signals (see [109] for a review). GSR tracks the electrical conductivity of the skin based on variations in moisture caused by sweating. The basic idea of GSR is that the sweat glands are controlled by the sympathetic nervous system (which modulates affect related flight or fight responses), so variations in moisture that are picked up by the GSR signal can reflect changes in physiological arousal. We note that even though here we focus exclusively on the GSR signal, our study is easily extensible to any of the recorded signals.

Each subject viewed each image for 10 seconds, with responses being recorded at a rate of 1,000 Hz. There was a 6-second break between image presentations in order to allow the signals to return to baseline values. A GSR signal of each subject
to each image was obtained, resulting in \(18 \times 89 = 1,602\) signals. The 89 signals for each subject were first standardized (converted to \(z\) scores) within the subject and then smoothed with a 0.3 Hz low-pass filter.

### 2.2.2 Network Construction

We construct networks from the collected physiological response data as follows. For each image, we create an unweighted, undirected network in which a node corresponds to a subject and an edge exists between two nodes if GSR responses of the corresponding subjects are “similar enough” with respect to a given similarity measure. Thus, each network structurally captures how different subjects respond to the same image as an initial step in modeling inter-individual variability.

To form a network, we need to choose: 1) a similarity measure for comparing GSR signals of two nodes, 2) a method for dealing with (e.g., filtering in some way) the resulting pairwise node similarity matrix, and 3) a strategy for defining from this matrix edges between the nodes (Figure 2.1). To obtain representative results, we evaluate the effect of the choice of these network construction parameters on our results. Namely, we test two similarity measures (Section 2.2.2.1). For each of the measures, we use two different strategies for dealing with the resulting matrices (Section 2.2.2.2). Finally, for each matrix, we define edges in the network in two ways (Section 2.2.2.3).

#### 2.2.2.1 Signal Similarity Measures

We measure GSR similarity by using: 1) Pearson correlation (PC) and 2) mutual information (MI). While PC measures linear relationships between signals, MI can capture more complex (non-linear) relationships between signals by relying on an information-theoretic notion of similarity \([45, 212]\). We normalize MI to take values between 0 and 1 \([45]\).
2.2.2.2 Dealing with Signal Similarity Matrices

For each image, we perform pairwise node comparison of the corresponding responses in order to obtain an $18 \times 18$ signal similarity matrix. Then, we either: 1) directly use this matrix to define edges (as described below), or 2) process this original matrix using the network deconvolution method \[57\] into a filtered matrix and then use the filtered matrix to define edges (as described below). The motivation for applying the network deconvolution method to the original matrix is as follows \[57\]. When constructing an observed network from an unfiltered matrix of pairwise node (i.e., subject) similarities (such as correlations), the resulting edges will likely include numerous indirect dependencies owing to transitive effects of the similarities. For example, if there is a strong dependency between subjects $X$ and $Y$ and between subjects $Y$ and $Z$ in the underlying true network, high similarities will likely also be visible between subjects $X$ and $Z$ in the observed network, thus inferring an edge from subjects $X$ to node $Z$ which might not actually exist in the true network. Moreover, even if this edge did exist in the true network, owing to additional indirect relationships between $X$ and $Z$, the strength of this edge may be overestimated; consequently, this true edge could be wrongly included into the observed network over some other true edge with higher initial strength but no indirect relationships to boost its strength up. Hence, filtering the matrix of node similarities by distinguishing between the convolved direct and indirect contributions could ensure that the resulting observed network matches the true network better than the observed network resulting from the non-filtered matrix of node similarities. As such, matrix filtering could lead to more meaningful results, and this is exactly what we examine. We refer to networks directly constructed from the original signal similarity matrices as unfiltered networks, and we refer to networks constructed from the filtered matrices as filtered networks.
2.2.2.3 Defining Edges

We define edges from a signal similarity matrix (unfiltered or filtered) as follows. We add an edge between two nodes if the similarity score between the corresponding subjects’ responses to the given image is above some threshold. We refer to the resulting networks as positive networks. In addition, we add an edge between two nodes if the absolute value of the similarity score is above some threshold. We refer to the resulting networks as positive-negative networks. Hence, in PC positive-negative networks, both subjects whose responses are strongly positively correlated as well as subjects whose responses are strongly negatively correlated are linked. Note that since MI values are nonnegative, positive-negative networks are constructed only with respect to PC.

To form a network (positive or positive-negative) from a signal similarity matrix (unfiltered or filtered), we need to specify an edge threshold value. We aim to choose this value in a way that keeps only statistically significant edges and provides a meaningful representation as well as interpretation of the data. Namely, for each image, we aim to construct a network that ideally links all 18 subjects, in order to include into the network as much of information from the data as possible. At the same time, we aim to construct a network that is not too dense, in order to mimic the sparse nature of many real-world networks as well as avoid randomness in network topology [145]. The choice of threshold value is likely to affect the resulting network topology: the higher the threshold, the smaller the number of nodes included into the network but the smaller the density; the lower the threshold, the larger the number of nodes included into the network but the larger the density. To determine an appropriate threshold, we analyze trends in network topology at multiple thresholds, as follows.

We begin by focusing on PC positive networks (Section 2.2.2). We vary PC threshold from 0.5 to 0.9 in increments of 0.1 and further from 0.9 to 0.99 in finer
increments of 0.01. We do not examine thresholds below 0.5, as the majority of all possible edges would already be included into networks at this threshold. For each examined threshold, we balance between the number of nodes included into networks and network density, as discussed above.

Empirically, we find that PC threshold of 0.95 results in the most appropriate networks, i.e., in networks that include many nodes while still being sparse enough. Also, since each GSR signal consists of $n = 1,000$ samples (Section 2.2.1), and since thus PC coefficient between two responses is computed by contrasting two 1,000-dimensional vectors, using a closed formula for statistical significance of PC at $n = 1,000$ [187], we find that threshold of 0.95 has a $p$-value of below $10^{-7}$. Thus, we adopt 0.95 as the threshold for PC positive networks, including $\sim 30\%$ of all possible edges into all 89 PC positive networks combined. (Not all networks necessarily have the same number of edges.) For a fair comparison, we adopt the same threshold for PC positive-negative networks.

However, the distribution of all possible PC scores and all possible MI scores is somewhat different. For example, for PC, $\sim 30\%$ of all possible edges are included into networks at threshold of 0.95, while for MI, $\sim 75\%$ of all edges are included into networks at the same threshold. Thus, for a fair comparison, instead of selecting the same threshold across the two different similarity measures, we adopt MI threshold of 0.975, as it also results in $\sim 30\%$ of all possible edges being included into MI positive networks.

2.2.2.4 Bottom Line: Network Types

To summarize, we have six different types of networks: PC positive, PC positive-negative, and MI positive, each with both an unfiltered and filtered version. For each of the six types, we generate 89 networks, one for each of the 89 images. Each network models similarity in responses between the 18 subjects and thus has 18 nodes.
However, since isolated nodes (i.e., nodes with no edges adjacent to them) do not contribute to the topology of the network, we remove such nodes.

2.2.3 Network Properties

To study whether network structure varies across the different network types, we analyze six popular concise “footprints” of network structure, or network properties: the number of nodes, number of edges, average degree, size of the largest connected component, diameter (see below), and the average clustering coefficient (see below). The diameter of a network is the maximum of shortest path lengths over all pairs of nodes in the network. The average clustering coefficient of a network is the average of clustering coefficients over all nodes in the network; the clustering coefficient of a node is the probability that two of the node’s neighbors are connected [131].

2.2.4 Clustering of Networks

After constructing the networks, we next ask whether networks corresponding to images of the same AV “ground truth” class (Section 2.2.1) are more “topologically similar” than networks corresponding to images from different classes. To answer this, we cluster the networks into non-overlapping groups based purely on their topological similarities, without using any “ground truth” knowledge about which network (i.e., image) corresponds to which “ground truth” class. In this way, we produce a network-based partition of the images. Then, we can compare such a partition with the AV “ground truth” partition (Section 2.2.1), in order to determine whether the two partitions significantly overlap. A significant overlap would indicate that based solely on network topological similarity we can group together images which group together according to the “ground truth”. (Note: we do not perform “graph clustering” of an individual network into groups (or communities) of nodes (or edges) [61, 194].) Instead, we consider each network as a separate “atomic” object and perform “data
clustering” of a *set of networks* into groups of “related” networks.

To cluster a set of objects (i.e., images), we need to define: 1) a measure of
distance (or equivalently, similarity) between the objects, 2) a clustering method, and
3) parameters of the method. We comprehensively test multiple network-based (as
well as non-network-based) distance measures (Section 2.2.4.1), clustering methods
(Section 2.2.4.2), and variations of the methods’ parameters (Section 2.2.4.2).

2.2.4.1 Distance Measures

We use *seven* network similarity measures: 1) *common edges*, i.e., the overlap of
the networks’ edge sets, as measured by Jaccard index ($|E_1 \cap E_2| / |E_1 \cup E_2|$),
where $E_1$ and $E_2$ are the two edge sets [179]; 2) absolute difference of the networks’
average clustering coefficients [103]; 3) absolute difference of their average diameters
(the average diameter of the network is the average of shortest path lengths over all node pairs [103]);
4) *Pearson correlation* of the networks’ degree distributions [103, 145]; 5) *Pearson correlation*
of their clustering spectra (the clustering spectrum of a network is the
distribution of average clustering coefficients of nodes with a particular degree) [103];
6) relative graphlet frequency distance (*RGF-distance*) (which compares frequencies
of all 3-5-node subgraphs, or *graphlets*, in two networks [165]); and 7) graphlet degree
distribution agreement (*GDD-agreement*) (which generalizes the degree distribution
into a spectrum of graphlet degree distributions [166]). We use *GraphCrunch* for all
comparisons [103, 131].

RGF-distance and GDD-agreement take into account more of network topology
compared to the other measures, and are thus considered to be more constrain-
ing measures of network similarity [134, 165, 166]. Intuitively, RGF-distance and
GDD-agreement are based on graphlets, which consider up to 4-deep network neigh-
borhood around each node. On the other hand, many other measures, e.g., the
clustering coefficient- and degree-based measures, consider only the direct, 1-deep
network neighborhood around a node \[103\] \[131\]. Also, RGF-distance and GDD-agreement compare distributions of statistics between two networks \[103\] \[131\], while many other measures, e.g., differences of average clustering coefficients or average diameters, compare a single statistic between the networks. Therefore, one would expect that RGF-distance and GDD-agreement would outperform the less constraining measures in terms of partition quality. Surprisingly, as we show in Section 2.3.1, this is not the case.

The common edges measure explicitly takes into account correspondence of node labels between networks, whereas all other measures are insensitive to label correspondence and consider only topological information. That is, for the common edges measure, similar topological patterns have to exist in two networks \textit{and} they have to exist between exactly the same nodes in the networks in order for the networks to be similar. On the other hand, for other measures, it is sufficient for similar topological patterns to exist in two networks \textit{in order for the networks to be similar}, without the explicit requirement for the patterns to exist between exactly the same nodes. So, one would expect that common edges would outperform other measures. Indeed, as we show in Section 2.3.1, this is what we observe in general.

Importantly, we want to ensure that we can obtain a more precise image partition by clustering network-based representations of the images via a network similarity measure than by clustering a non-network representation of the image data via some statistical, non-network-based image similarity measure. That is, we want to ensure that it is indeed beneficial to use network analysis to study physiological data. For this purpose, we define an additional non-network-based measure of similarity between two images, as follows. For each image and for each similarity measure (i.e., PC or MI; Section 2.2.2), we construct a vector with \(\begin{pmatrix} 18 \\ 2 \end{pmatrix}\) values, where each value corresponds to similarity between responses of two corresponding subjects to the given image. Now, instead of selecting a similarity threshold to define a subset of the \(\begin{pmatrix} 18 \\ 2 \end{pmatrix}\)
values as edges in a network corresponding to the given image and then computing similarity between two images by comparing their networks, as above, here, to compute similarity between two images, we directly compare two \( \binom{18}{2} \)-dimensional vectors corresponding to two images using Pearson correlation. We refer to this measure as the non-network (NON) distance.

Thus, we consider the total of eight distance measures. For each, we construct matrices of pairwise image distances and input these matrices into a clustering method, in order to group (separate) similar (dissimilar) images.

### 2.2.4.2 Clustering Methods and Their Parameters

To test how the choice of clustering method affects a partition quality, we use two clustering methods: 1) hierarchical clustering (HIE) and 2) \( k \)-medoids clustering (KM) \([61]\). Parameters of HIE are: the linkage method, i.e., the way of measuring the distance between two clusters, and the choice of how to cut a hierarchical tree (or dendrogram). Clearly, different values of each parameter could lead to different partitions. Thus, we test four linkage methods: single, complete, average, and weighted \([61]\). To cut a dendrogram, we specify the desired number of clusters, \( k \), testing all possible values of \( k \), from 1 to 89, in increments of 1. Note that we also tested other strategies for cutting a dendrogram (e.g., by specifying the maximum allowed inter-cluster distance), but their performance in terms of the partition quality was inferior (results not shown). KM is a modification of \( k \)-means clustering that requires using actual data points (in our case, networks) as cluster centers, rather than allowing centers to be non-data points, as \( k \)-means does \([93]\). And since our clustering distance measures require centers to be data points, using \( k \)-means is inappropriate, or in other words, \( k \)-medoids needs to be used. The only parameter of KM is the desired number of clusters, \( k \), for which we again test all 89 possible values. Thus, we test four linkages for HIE and all possible values of the desired number of clusters, \( k \), for
both HIE and KM \((k = 1, 2, ..., 89)\).

All combinations of the two signal similarity measures (PC and MI), two ways to deal with signal similarity matrices (using either the original unfiltered matrix or the filtered matrix), two ways to define edges (positive and positive-negative), eight distance measures (seven network-based measures and NON-distance), and two clustering methods (HIE and KM) lead to 96 different parameter combinations: 48 combinations for unfiltered networks and 48 combinations for filtered networks. Actually, since for unfiltered networks, NON-distance produces the same distance matrix for PC positive networks and PC positive-negative networks, and since it does so for both hierarchical and \(k\)-means clustering, the actual number of different combinations for unfiltered networks is 46 instead of 48. Thus, in total, over both unfiltered and filtered networks, there are \(46 + 48 = 94\) different parameter combinations. For each of these combinations, we choose the “best” partition over all combinations of clustering parameters (over all possible choices of \(k\) and linkage for HIE, and over all possible choices of \(k\) for KM). By “best”, we mean the most significant according to the criterion introduces in the following section.

2.2.5 Evaluating Partition Quality

Upon producing a partition of images, we evaluate the quality of the partition with respect to its overlap with the “ground truth” knowledge about the images. Namely, we evaluate a partition: 1) by comparing it against the AV “ground truth” partition, and 2) according to its semantic meaning.

To compare two partitions, we use \textit{Adjusted for chance Information Distance} (AID) measure \cite{211, 212}. This measure uses notions of entropy and MI to determine the similarity between two partitions from an information theoretic perspective. It quantifies how much knowing one of the partitions reduces uncertainty about the other \cite{212}. The lower the AID value, the more similar two partitions. AID already
incorporates “adjustment for chance” that allows for comparing partitions of different cluster sizes without bias [211]. As a consequence, AID gives a way to rank pairs of partitions based on their similarities. This is very useful in our study, because it allows us to evaluate the fit of the AV “ground truth” partition to many different partitions resulting from the different clustering strategies. And by comparing multiple partitions to the AV partition, we can determine which one of them is better, i.e., closer to the “ground truth”. We determine the statistical significance of our AID score between two partitions empirically, as the percentage of $10^6$ randomly generated partitions that have the same or better AID scores than our actual AID score. This percentage is our $p$-value. If an AID score between some partition and the AV “ground truth” partition has a $p$-value below 0.01, we refer to that partition as “statistically significant”. For further analysis, among all 94 best partitions (corresponding to 94 different parameter combinations), we focus only on those that are statistically significantly similar to the AV partition with respect to AID.

In addition to comparing our “statistically significant” partitions with the AV “ground truth” partition, we also assess them using latent semantic analysis (LSA) [107]. LSA is a statistical technique that computes the conceptual similarity of two texts (words, sentences, or documents) by leveraging second-order co-occurrence relationships from large text corpora. For each pair of our images, we derive LSA similarities between high-level image names (e.g., kittens, garbage, spider) depicted in the images using the online LSA tools[^1]. That is, we obtain an additional “ground truth” data set based purely on high-level semantic meanings of the images. To evaluate the quality of a partition from the LSA perspective, we compare intra- and inter-cluster LSA similarities within this partition. In a high quality partition, intra-cluster similarities would be statistically significantly higher than inter-cluster similarities. To evaluate the statistical significance of the difference between intra- and inter-cluster similarities.

[^1]: [http://lsa.colorado.edu/](http://lsa.colorado.edu/)
similarities in a partition, we compare the vector of all pairwise intra-cluster similarities with the vector of all pairwise inter-cluster similarities using the Wilcoxon rank-sum test [187]. By doing this for multiple partitions, and by comparing the resulting p-values across the partitions, we can determine which of the partitions is more significant and thus more semantically meaningful. We do this not only for our network-based partitions but also for the AV “ground truth” partition (which might be noisy; Section 2.2.1), in order to evaluate whether any of our network-based partitions improve upon the AV partition with respect to LSA. We note that by using high-level labels to categorize each image, we rely on somewhat primitive keywords of images (e.g., dog) instead of more informative descriptions (e.g., dog stretching out in the lawn gazing at mail man). We do this because our primary goal is to capture primitive semantic influences on physiological early responding triggered by the “gist” of the image rather than more complex cognitive appraisals of the information depicted in each image.

2.3 Results and Discussion

We first discuss results for unfiltered networks (Section 2.3.1), followed by results for filtered networks (Section 2.3.2), in order to evaluate the effect of filtering on the quality of the results (Section 2.3.3). Also, we discuss results and implications of a representative network-based partition (Section 2.3.4).

2.3.1 Results for Unfiltered Networks

Here, we summarize topological properties of unfiltered networks (Section 2.3.1.1) as well as the quality of their partitions (Section 2.3.1.2).
2.3.1.1 Network Topological Trends

We study six topological properties of the networks of different types: the number of nodes, number of edges, size of the largest connected component, maximum diameter, average clustering coefficient, and average degree (Section 2.2.3). Since topological trends are very similar for PC positive and PC positive-negative networks, except that PC positive-networks are slightly denser, in this section, we illustrate results for PC positive networks only. But all results hold for PC positive-negative networks as well. Then, we contrast results for PC positive networks against results for MI positive networks.

While 69% of all edges in unfiltered networks are common to all three network types (Figure 2.3(a)), there is still a notable variability in the topology of different networks types, which could affect network-based clustering of images. Namely, PC networks tend to have more nodes than MI positive networks, while MI positive networks tend to be denser than PC networks (Figure 2.4). Also, with respect to the number of edges, in PC networks, different networks of a given size (in terms of the number of nodes) have very different numbers of edges (Figure 2.4(a)). On the other hand, in MI positive networks, different networks of a given size tend to agree more
on their numbers of edges (Figure 2.4(b)). Similar trends are observed with respect to the average degree (results not shown). Further, largest connected components of PC networks of a given size tend to be smaller than components of MI networks of the same size (Figures 2.4(c) and 2.4(d)). That is, PC networks typically contain multiple connected components (Figure 2.4(c)), while in almost each MI network, all of the nodes are contained within the network’s largest (and thus only) connected component (Figure 2.4(d)).

Networks of different types have similar trends with respect to their diameters and average clustering coefficients. Diameters of all networks are relatively small and their clustering coefficients are relatively high. This is encouraging, as the observed behavior is typical for many real-world networks [145].

2.3.1.2 Quality of Partitions

After we construct networks, we use them to partition the image set (Section 2.2.4). We focus only on “statistically significant” partitions (Section 2.2.5). There are 17 such partitions for unfiltered networks, out of 46 possible partitions (Table 2.2). When we evaluate the effect of different network construction and clustering parameters on the partition quality, we find that the choice of the parameters affects the resulting partitions. For example, we find that PC is generally better (i.e., it produces more “statistically significant” partitions) than MI, with PC positive-negative networks demonstrating the best results (Table 2.1). Regarding the choice of distance measure, common edges and the difference of average clustering coefficients are superior, contrary to our expectation that more constraining measures of network similarity, such as GDD-agreement and RGF-distance, would be the best. Importantly, we find that there is no single choice for any of the parameters that works for all combinations of the other parameters. This is especially true for HIE. This implies that we still have to consider all “statistically significant” partitions and choose the
Figure 2.4: Illustration of topological properties of unfiltered networks. We plot the distribution of edge counts over (a) PC positive and (b) MI positive networks with the given number of nodes (or vertices) (as shown on x-axis). Also, we plot the distribution of the sizes of the largest connected components (in terms of the number of nodes) over (c) PC positive and (d) MI positive networks with the given number of nodes (as shown on x-axis). In each panel, the larger the number of networks with given properties, the larger the size of the corresponding circle. Also, in each panel, green and red crosses correspond to theoretical minimum and maximum values of given properties, respectively. For example, in a network with \( n \) non-isolated nodes, there has to be at least \( \left\lceil \frac{n}{2} \right\rceil \) edges (colored in green in panels (a) and (b)) and there can be at most \( \binom{n}{2} \) edges (colored in red in panels (a) and (b)). Or, in a network with \( n \) non-isolated nodes where \( n \geq 2 \), the largest connected component has to have at least 2 nodes (colored in green in panels (c) and (d)) and it can have at most \( n \) nodes (colored in red in panels (c) and (d)).
best according to a desired criterion.

The fact that we are able to construct “statistically significant” partitions using a network-based approach implies that differences in physiological response patterns of subjects to various images captured by our approach are meaningful with respect to the AV “ground truth” partition. Also, the network-based partitions tend to fit the AV partition better than the non-network-based partitions (Table 2.2), indicating that network analysis indeed can improve interpretation of the data.

To further evaluate the quality of the “statistically significant” partitions, we measure their semantic meaningfulness using LSA (Section 2.2.5). Then, we ask whether our network-based partitions outperform in terms of LSA: 1) the AV partition and 2) the non-network-based partitions. If so, that would mean that: 1) even though our partitions do not perfectly match the AV partition (but are still statistically significantly similar to it), they are more semantically meaningful (according to LSA), and 2) they are more meaningful than the non-network based analysis of the same physiological data that we employed in our study. That is, this would further confirm the validity of our network analysis strategy in the context of affective physiological data.

Indeed, this is what we observe (Table 2.2). While the LSA $p$-value for the AV partition is 0.261 (and thus non-significant), five of our partitions (four for HIE and one for KM) are semantically meaningful in terms of LSA. Importantly, all of them are network-based. Even though non-network-based partitions are “statistically significant”, none of them is semantically meaningful at the same time. This confirms that network analysis can improve interpretation of physiological data by capturing both AV “ground truth” and semantics.
<table>
<thead>
<tr>
<th>Distance measure</th>
<th>Linkage method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
</tr>
<tr>
<td>NON distance</td>
<td>PC</td>
</tr>
<tr>
<td>Common edges</td>
<td>PC(_p), MI(_p)</td>
</tr>
<tr>
<td>GDD-agreement</td>
<td>PC(_p)</td>
</tr>
<tr>
<td>RGF-distance</td>
<td>PC(_pn)</td>
</tr>
<tr>
<td>Pearson correlation of degree distributions</td>
<td>PC(_pn)</td>
</tr>
<tr>
<td>Pearson correlation of clustering spectra</td>
<td>-</td>
</tr>
<tr>
<td>Difference of average diameters</td>
<td>MI(_p)</td>
</tr>
<tr>
<td>Difference of average clustering coefficients</td>
<td>PC(_pn)</td>
</tr>
</tbody>
</table>

* Clearly, none of the combinations works for all three network types. Similar results hold when fixing network types and linkage methods: none of their combinations works for all distance measures. Also, similar results hold when fixing network types and distance measures: none of their combinations works for all linkages.*
**TABLE 2.2**

QUALITY IN TERMS OF AID $p$-VALUES AND LSA $p$-VALUES OF THE “STATISTICALLY SIGNIFICANT” IMAGE PARTITIONS RESULTING FROM CLUSTERING OF *UNFILTERED* NETWORKS

<table>
<thead>
<tr>
<th>#</th>
<th>Partition description</th>
<th>Network type</th>
<th>Clust. method</th>
<th>Distance measure</th>
<th>Unfiltered networks</th>
<th>Filtered networks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AID $p$-value</td>
<td>LSA $p$-value</td>
</tr>
<tr>
<td>1</td>
<td>PC positive</td>
<td>HIE</td>
<td>Common edges</td>
<td></td>
<td>3.70E-05</td>
<td>0.023</td>
</tr>
<tr>
<td>2</td>
<td>PC, non-network-based</td>
<td>HIE</td>
<td>NON-distance</td>
<td></td>
<td>1.15E-03</td>
<td>0.232</td>
</tr>
<tr>
<td>3</td>
<td>MI positive</td>
<td>HIE</td>
<td>Common edges</td>
<td></td>
<td>1.25E-03</td>
<td>1.00E-03</td>
</tr>
<tr>
<td>4</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>Diff. of avg. clust. coeff.</td>
<td>2.16E-03</td>
<td>1.72E-10</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>PC of degree distr.</td>
<td>3.25E-03</td>
<td>9.58E-07</td>
<td>34</td>
</tr>
<tr>
<td>6</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>RGF-distance</td>
<td></td>
<td>5.68E-03</td>
<td>0.013</td>
</tr>
<tr>
<td>7</td>
<td>MI positive</td>
<td>HIE</td>
<td>Diff. of avg. diameters</td>
<td>6.40E-03</td>
<td>1.04E-05</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>PC positive</td>
<td>HIE</td>
<td>Diff. of avg. clust. coeff.</td>
<td>7.47E-03</td>
<td>0.203</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>PC, non-network-based</td>
<td>KM</td>
<td>NON-distance</td>
<td></td>
<td>1.51E-04</td>
<td>0.123</td>
</tr>
<tr>
<td>10</td>
<td>PC positive</td>
<td>KM</td>
<td>Diff. of avg. clust. coeff.</td>
<td>3.40E-04</td>
<td>8.00E-03</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>PC of degree distr.</td>
<td>1.94E-03</td>
<td>0.246</td>
<td>44</td>
</tr>
<tr>
<td>12</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Common edges</td>
<td></td>
<td>2.00E-03</td>
<td>0.051</td>
</tr>
<tr>
<td>13</td>
<td>PC positive</td>
<td>KM</td>
<td>Common edges</td>
<td></td>
<td>2.66E-03</td>
<td>0.065</td>
</tr>
<tr>
<td>14</td>
<td>MI positive</td>
<td>KM</td>
<td>Common edges</td>
<td></td>
<td>2.93E-03</td>
<td>0.177</td>
</tr>
<tr>
<td>15</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Diff. of avg. clust. coeff.</td>
<td>3.50E-03</td>
<td>0.416</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>MI, non-network-based</td>
<td>KM</td>
<td>NON-distance</td>
<td></td>
<td>4.69E-03</td>
<td>0.403</td>
</tr>
<tr>
<td>17</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Diff. of avg. diameters</td>
<td>5.53E-03</td>
<td>0.526</td>
<td>10</td>
</tr>
</tbody>
</table>

*Network construction and clustering parameters are listed under “Partition description.” |C| denotes the number of clusters in the corresponding partition. The lower the AID $p$-value, the more similar the partition is to the AV “ground truth” partition. The lower the LSA $p$-value, the more semantically meaningful the partition. For each clustering method (HIE and KM), partitions are listed in the increasing order of their AID $p$-values corresponding to unfiltered networks. The three rightmost columns show AID $p$-values, LSA $p$-values, and |C| of the equivalent partitions resulting from the same network construction and clustering parameters but from filtered networks instead of unfiltered ones. These partitions of filtered networks are not necessarily “statistically significant.” All AID and LSA $p$-values (corresponding to both unfiltered and filtered networks) lower than 0.01 are shown in bold.*

31
<table>
<thead>
<tr>
<th>#</th>
<th>Partition description</th>
<th>Network type</th>
<th>Clust. method</th>
<th>Distance measure</th>
<th>AID p-value</th>
<th>LSA p-value</th>
<th>LSA p-value</th>
<th>AID p-value</th>
<th>LSA p-value</th>
<th>LSA p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MI positive</td>
<td>HIE</td>
<td>GDD-agreement</td>
<td>1.80E-04</td>
<td>0.067</td>
<td>0.137</td>
<td>5.00E-03</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PC positive</td>
<td>HIE</td>
<td>RGF-distance</td>
<td>2.99E-04</td>
<td>0.680</td>
<td>0.034</td>
<td>0.034</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PC positive</td>
<td>HIE</td>
<td>PC of clust. spectra</td>
<td>7.46E-04</td>
<td>0.310</td>
<td>0.021</td>
<td>0.290</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MI positive</td>
<td>HIE</td>
<td>RGF-distance</td>
<td>1.41E-03</td>
<td>0.845</td>
<td>0.030</td>
<td>0.517</td>
<td>57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>Diff. of avg. clust. coeff.</td>
<td>4.81E-03</td>
<td>0.029</td>
<td>26</td>
<td>7.47E-03</td>
<td>0.203</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>GDD-agreement</td>
<td>5.68E-03</td>
<td>1.90E-09</td>
<td>0.228</td>
<td>0.090</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PC positive</td>
<td>HIE</td>
<td>PC of degree distr.</td>
<td>6.15E-03</td>
<td>0.908</td>
<td>0.008</td>
<td>0.014</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PC positive</td>
<td>HIE</td>
<td>Diff. of avg. clust. coeff.</td>
<td>6.82E-03</td>
<td>0.214</td>
<td>11</td>
<td>0.054</td>
<td>0.835</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PC, non-network-based</td>
<td>HIE</td>
<td>NON-distance</td>
<td>8.50E-03</td>
<td>0.017</td>
<td>7</td>
<td>2.24E-03</td>
<td>0.379</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>Diff. of avg. diameters</td>
<td>9.27E-03</td>
<td>0.041</td>
<td>30</td>
<td>0.155</td>
<td>0.760</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PC positive-negative</td>
<td>HIE</td>
<td>RGF-distance</td>
<td>9.46E-03</td>
<td>1.00E-03</td>
<td>18</td>
<td>5.68E-03</td>
<td>0.309</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>GDD-agreement</td>
<td>1.12E-04</td>
<td>0.684</td>
<td>16</td>
<td>0.026</td>
<td>0.721</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>PC positive</td>
<td>KM</td>
<td>GDD-agreement</td>
<td>6.99E-04</td>
<td>0.114</td>
<td>18</td>
<td>0.021</td>
<td>0.324</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Common edges</td>
<td>1.29E-03</td>
<td>0.042</td>
<td>7</td>
<td>2.00E-03</td>
<td>0.054</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>MI, non-network-based</td>
<td>KM</td>
<td>NON-distance</td>
<td>4.02E-03</td>
<td>0.815</td>
<td>20</td>
<td>4.69E-03</td>
<td>0.403</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>PC positive</td>
<td>KM</td>
<td>PC of clust. spectra</td>
<td>4.51E-03</td>
<td>0.159</td>
<td>19</td>
<td>0.038</td>
<td>0.664</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MI positive</td>
<td>KM</td>
<td>GDD-agreement</td>
<td>5.46E-03</td>
<td>1.00E-03</td>
<td>21</td>
<td>0.160</td>
<td>1.02E-07</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Diff. of avg. diameters</td>
<td>5.76E-03</td>
<td>0.058</td>
<td>26</td>
<td>5.53E-03</td>
<td>0.526</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>PC positive</td>
<td>KM</td>
<td>Diff. of avg. clust. coeff.</td>
<td>5.95E-03</td>
<td>0.823</td>
<td>11</td>
<td>3.40E-04</td>
<td>8.00E-03</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>PC positive-negative</td>
<td>KM</td>
<td>Diff. of avg. clust. coeff.</td>
<td>7.12E-03</td>
<td>0.075</td>
<td>12</td>
<td>3.50E-03</td>
<td>0.416</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>MI positive</td>
<td>KM</td>
<td>Common edges</td>
<td>7.16E-03</td>
<td>0.279</td>
<td>11</td>
<td>2.93E-03</td>
<td>0.177</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

*Column descriptions are the same as in Table 2.2.*
2.3.2 Results for Filtered Networks

Next, we show topological (Section 2.3.2.1) and partition quality (Section 2.3.2.2) results for filtered networks.

2.3.2.1 Network Topological Trends

We find that the topology of filtered networks is extremely sensitive to the choice of network construction strategy: only 6% of all edges in filtered networks (compared to 69% for unfiltered networks) are shared among the three network types (Figure 2.3(b)). Importantly, even within a single network type, there is a notable variability in most of the analyzed network topological properties, which could have drastic effects on network-based clustering.

2.3.2.2 Quality of Partitions

There are 21 “statistically significant” partitions of filtered networks, out of 48 possible partitions (Table 2.3). When we evaluate the effect of different network construction and clustering parameters on the partition quality, just as for unfiltered networks, we again find that the choice of the parameters affects the resulting partitions. But unlike for unfiltered networks, we find that more topologically constraining network distance measures (Section 2.2.4.1) are now superior, as we would expect: RGF-distance results in “statistically significant” partitions for all three network types in case of HIE, and GDD-agreement does the same for all three network types in case of KM.

Although it is possible to obtain “statistically significant” partitions from filtered networks using the NON-distance measure, importantly, AID scores of these partitions are worse than those of network-based partitions (Table 2.3). That is, the network-based partitions again tend to fit the AV partition better than the non-network-based partitions.
When we evaluate the “statistically significant” partitions in terms of LSA, we find that they are more similar than AV “ground truth” partition (Table 2.3): three of the partitions are semantically meaningful with p-value of 0.01 (recall that the LSA p-value for the AV “ground truth” partition is 0.261 and is thus non-significant). Importantly, just as for unfiltered networks, none of the non-network-based partitions for filtered networks is semantically meaningful at the same time. However, unlike for unfiltered networks, we note that GDD-agreement, which we would expect to be superior to all other distance measures, is indeed superior for filtered networks, as it is the only measure that yields to semantically meaningful partitions for both clustering methods (HIE and KM) as well as for both signal similarity measures (PC and MI) (Table 2.3).

In summary, just as for unfiltered networks, clustering of filtered networks produces partitions that are statistically significantly similar to the AV “ground truth” partition while fitting more closely to LSA than the AV “ground truth” partition. Importantly, network-based partitions again lead to better results than non-network-based partitions, further confirming that network analysis can improve the interpretation of physiological data.

2.3.3 Comparison of Results for Unfiltered and Filtered Networks

Here, we contrast topological properties between unfiltered and filtered networks (Section 2.3.3.1). Then, we ask which of the two should be used from a practical point of view (Section 2.3.3.2).

2.3.3.1 Network Topological Trends

Even though we designed our study so that all network construction strategies result in the same average network density (Section 2.2.2), unfiltered and filtered networks differ in their topology. To start with, while the three network types share
Figure 2.5: Illustration of differences in topological properties of unfiltered and filtered networks. We plot the distribution of edge counts over MI positive networks with the given number of nodes (as shown on x-axis), constructed from: [a] unfiltered and [b] filtered signal similarity matrices. The larger the number of networks with the given property (i.e., the given number of nodes and edges), the larger the size of the corresponding circle. Green and red “×” marks in the panels correspond to theoretical minimum and maximum values of the illustrated properties, respectively.

The majority of all edges in unfiltered networks (Figure 2.3(a)), they share only a small portion of all edges in filtered networks (Figure 2.3(b)). Also, when we ignore the network types and look at all edges in both unfiltered and filtered networks, only 44% of the edges are common to the two (Figure 2.3(c)).

Going beyond simply counting the edges in the intersection between different network types, in general, unfiltered and filtered networks also have different topological properties. Namely, they differ in terms of network sizes (Figure 2.5). Also, filtered networks have smaller clustering coefficients (or, intuitively, fewer triangles) than unfiltered networks. This could be because the network deconvolution filtering method is likely to decrease the effect of “spurious” pairwise node similarities [57], and thus, potential edges contributing to indirect paths (such as the indirect path between two nodes in a triangle) could be weighted lower and consequently removed from the network.
2.3.3.2 Unfiltered vs. Filtered Networks: Which One to Use from Practical Point of View?

Clearly, the network deconvolution method alters the topology of resulting filtered networks compared the topology of unfiltered networks. Still, both unfiltered and filtered networks produce statistically significant and semantically meaningful results (Tables 2.2 and 2.3, respectively). Yet, results that are significant and meaningful for unfiltered networks are not necessarily significant and meaningful for filtered networks (Table 2.2), and vice versa (Table 2.3). Thus, the question is which one to use: unfiltered or filtered networks? To answer this, we first contrast the two by measuring which one shows significant correlation of their network-based image distances with “ground truth” image distances, before partitioning the images based on their network-based distances (this allows for a direct comparison of unfiltered and filtered networks without being confounded by clustering). Then, we contrast unfiltered and filtered networks with respect to the quality of their partitions produced based on the network-based image distances.

**Correlation with “ground truth” image distances.** We directly correlate each of (non-)network-based image distance matrices with each of four “ground truth” image distance matrices: 1) Arousal (comparing IAPS normative arousal scores of the images), 2) Valence (comparing IAPS normative valence scores of the images), 3) Arousal-Valence (comparing the combination of the previous two scores; Figure 2.2(a)), and 4) LSA (comparing semantic similarity of the images). We do this via Mantel test, which is convenient for computing the significance of correlation between two distance matrices of related objects after adjusting for item-dependence [123].

We find that NON-distance, the non-network-based approach, captures (i.e., significantly correlates with) Arousal, but nothing else. And capturing Arousal is expected for GSR as repeatedly documented in the literature (e.g., [6, 109]). Thus, the
A non-network-based approach does not seem to uncover any interesting (i.e., unexpected) signal from the data. Network-based approaches, on the other hand, capture all four types of the “ground truth” image distance matrices. That is, network-based approaches go beyond simply capturing the expected Arousal – they also significantly correlate with distance matrices of Valence, Arousal-Valence, and semantics (LSA). Hence, we again confirm that network analysis of the data has important practical applications, as it can capture interesting results that can easily be missed by non-network-based analysis.

Going back to determining which one is more efficient, unfiltered or filtered networks, we find that the latter outperform (in terms of the significance of correlations) the former for three out of the four “ground truth” image distance matrices: Arousal, Valence, and LSA. While both types of networks capture (i.e., significantly correlate with) Arousal and LSA, only unfiltered networks capture Arousal-Valence, and only filtered networks capture Valence. This suggests that perhaps the most defensible approach is to consider both filtered and unfiltered networks, though they may be added advantages to filtered networks as discussed below.

**Quality of partitions.** Unfiltered and filtered networks show comparable performance in terms of partition quality (Tables 2.2 and 2.3), although filtering leads to a slightly larger number of statistically significant partitions (21 for filtered networks vs. 17 for unfiltered networks) as well as semantically meaningful partitions (seven for filtered networks vs. five for unfiltered networks). However, the partition with the lowest AID score was obtained for a partition of unfiltered networks (partition 1 in Table 2.2).

We find that topologically more constraining network distance measures, such as GDD-agreement and RGF-distance, which we would expect to be superior, are indeed superior for filtered networks. For unfiltered networks, only one “statistically significant” partition is obtained by using these measures, while there are eight such
partitions for filtered networks (three of which are also semantically meaningful). Moreover, for filtered networks, both the partition with the lowest AID score for HIE, as well as the partition with the lowest AID score for KM (partitions 1 and 12 in Table 2.3 respectively) result from GDD-agreement. And since we expect GDD-agreement and RGF-distance to work better than their topologically simpler counterparts, these results might imply that filtered networks are preferred, at least from this perspective.

We also study the robustness of our approach to network filtering. That is, we check whether the same combinations of network construction and clustering parameters that result in “statistically significant” partitions for unfiltered networks also result in “statistically significant” partitions for filtered matrices (the last two columns of Table 2.2), and vice versa (the last two columns of Table 2.3). In this context, we find that HIE is less robust than KM. Specifically, for HIE, only one out of eight “statistically significant” unfiltered partitions remains “statistically significant” after filtering (Table 2.2), and only three out of eleven “statistically significant” filtered partitions remain statistically significant without filtering (Table 2.3). On the other hand, for KM, these results are six out of nine and six out of ten, respectively. Importantly, there is no partition that is both “statistically significant” and semantically meaningful both with and without filtering.

**Unfiltered vs. filtered networks: summary.** Our results suggest that filtering of similarity matrices can significantly affect the topology of the resulting networks and the quality of the resulting partitions. Importantly, network construction and clustering strategies that work well for unfiltered networks do not necessarily work well for filtered network (and vice versa). Although filtering does not drastically improve the quality of the image partitions, it leads to a slightly larger number of “statistically significant” and semantically meaningful partitions. Filtered networks also correlate more significantly than unfiltered networks with three out of four sets of
“ground truth” image distances. Furthermore, filtering works better in combination with more topologically constraining distance measures, as we would expect, which may imply that filtered networks have less noise compared to unfiltered networks.

2.3.4 In-depth Analysis and Interpretation of a Representative Network-based Image Partition

We now focus on a representative “statistically significant” and semantically meaningful network-based partition for further in-depth analysis. In particular, we select partition 10 from Table 2.2. We choose this partition out of all partitions from Tables 2.2 and 2.3 as it mimics the closest the AV “ground truth” partition in terms of the number of clusters (both partitions have nine clusters) as well as cluster sizes. As such, it enables a fair interpretation of its results with respect to the AV “ground truth.”

To analyze this partition, we compute for each cluster its average valence score as well as its average arousal score over all images in the cluster, after assigning each image the score of 0, 0.5, or 1 for its low, medium, or high arousal as well as for negative, neutral, and positive valence, respectively. Figure 2.6 depicts these averages for each cluster on a two-dimensional arousal-valence space, while Figure 2.7 visualizes the membership of images in each cluster. Ideally, we would hope for considerable similarities in average arousal-valence scores within each cluster and considerable differences in the scores across the clusters, as this would validate the correctness of our approach.

Indeed, this is what we observe (Figures 2.6 and 2.7). Clusters 2 and 7 contain images with negative valence, such as garbage, vomit, trash in C2 and a cockroach, surgery, and a gun in C7. They also contain images with neutral valence, such as a mask and a dental exam in C2 and an electric outlet and men in C7. In general, mean valence scores for these clusters are highly similar (0.28 for C2 and 0.25 for
C7), but these clusters can be distinguished via mean arousal values (0.33 for C2 and 0.5 for C7), with C7 containing images that are more emotionally arousing than C2. With one exception, neither cluster contains any positively valenced images.

Conversely, clusters C8 and C9 contain mostly positive-valenced with occasional neutral-valenced images, but they rarely contain negatively valenced images; mean valence scores are 0.64 and 0.67 for C8 and C8, respectively. Example images in these clusters include a couple, a watermelon, a tomato in C8 and a rabbit and surfers in C9. Once again, arousal discriminates these clusters, with images in C9 (e.g., surfers) being more arousing than images in C8 (e.g., tomato); mean arousal score in C9 of 0.50 is double that in C8 of 0.23.

Clusters C1 and C5 occupy similar positions in the arousal-valence space with mean arousal scores of 0.71 and 0.73 and mean valence scores of 0.62 and 0.59, respectively. Sample images in these clusters include fireworks, money, gold, and clowns (which are arousing and have positive valence) along with surgery, a roach on a slice of pizza, and a starving child (which are arousing and have negative valence). Therefore, clusters C1 and C5 seem to capture physiological responses to arousal rather than valence.

Conversely, clusters C4 and C6 occupy similar positions in the arousal-valence space, but there are some marked differences among these clusters. Most notable is the fact that C4 contains an equal number of positively-valenced (e.g., a butterfly or a baby) and negatively-valenced images (e.g., a crying family or skulls), which leads to a mean neutral-valence profile (with score of 0.44). On the other hand, although C6 has a similar mean valence score of 0.5, it mainly contains neutral images (e.g., a bicyclist, cheerleaders, or a pig), rather than a mix of positively- and negatively-valenced images. Additionally, even though average arousal values for these clusters are also similar, C6 is slightly more arousing (with mean arousal score of 0.39) than C4 (with mean arousal score of 0.28).
Finally, while cluster C3 is consistent with average neutral valence (with average valence of 0.44, containing neutral images of e.g., a building, an actor, a man, or a skyscraper), this cluster is more interesting in terms of the variability in arousal that it captures, since it contains highly arousing (e.g., erotic female), medium arousing (e.g., actor), and low arousing (e.g., a man) images, resulting in average medium-high arousal (with score of 0.61).

A detailed analysis of this exemplary partition unveils some interesting insights pertaining to physiological responses to affective stimuli. An interesting observation is that the physiological responses seem to compress the arousal-valence space rather than being uniformly distributed across the space (Figure 2.6). This is because the clusters rarely adhere to the expected AV “ground truth” in that there is rarely a cluster with images that perfectly map onto the AV “ground truth” image classes. For example, there is no high-arousal positive-valence cluster in our partition. Instead, the clusters contain blends of images that are mostly similar in valence but differ in arousal (e.g., C6), that are mostly similar in arousal but differ in valence (e.g., C5, C1), or that differ along both dimensions (e.g., C4). Given that our networks are created based on similarity of participants’ physiological responses to images, our results suggest that inter-individual physiological responding does not neatly align with the expected AV “ground truth,” but is not entirely random either, as evident from the observed patterns in our clusters. We suspect that the “meaning” of the images might also play a role in how individuals physiologically respond to them. Hence, both the affective dimension (i.e., valence and arousal) and the cognitive dimension (i.e., meaning) might be needed to explain variability in physiological responding. This is particularly encouraging since we have shown that our network-based approach captures both of these aspects, since it significantly aligns with both the AV “ground truth” and LSA (meaning) partitions.
Figure 2.6: Arousal-valence scores of different clusters in the representative network-based partition, where the scores are averaged over all images in the given cluster.

Figure 2.7: Membership of the 89 images in different clusters of the representative network-based partition. Each image is depicted as a circle whose color corresponds to its IAPS normative valence score (red – negative valence, yellow – neutral valence, or green – positive valence) and whose size corresponds to its IAPS normative arousal score (small – low arousal, medium – medium arousal, or large – high arousal).
2.4 Conclusions

We use a network approach to study affective physiological data. Namely, we model images as networks and group images with similar network topologies. We perform a systematic analysis of the effect of different network construction and clustering approaches, concluding that the each choice can affect the results. For network inference and clustering communities, this highlights the importance of considering various strategies. Nonetheless, we show that via network analysis we can construct image partitions that are significantly similar to the AV “ground truth” partition, while at the same time yield deeper insights by also being sensitive to semantics as estimated by LSA. Importantly, we show that such a result cannot be obtained via a non-network-based analysis of the same data. Thus, viewing affective physiological data through a network lens can yield deeper insights by improving analysis of the data.

We introduce a framework for systematic network analysis of human physiological responses. There are several future extensions of our research. A next step could be to identify communities of individuals with respect to their (dis)similarities in physiological responses to affective stimuli and to investigate whether these communities can be discriminated on the basis of trait-based individual differences (e.g., personality factors, distress tolerance, generalized anxiety, etc.). Also, while our data is based on 18 subjects, studying more subjects would make the results more generalizable. While we construct image networks modeling similarities between responses of different individuals to a given image, one can also construct subject networks, modeling similarities between responses of a given subject to different images, which could give complementary insights. Finally, while we focus on GSR signals, our framework can be applied to other signals, e.g., ECG or EMG, which would allow to study relationships between different physiological channels. In this context, the different signals could be studied individually from network perspective and then their results could
be integrated for a more comprehensive understanding, or the networks corresponding to the different signals could be integrated first, prior to any network analysis; how exactly this should be done is the subject of future research, as it falls under the umbrella of the field of heterogeneous (network) data analysis, which is somewhat in its infancy. All of the above future extensions of our current work could lead to the ultimate goal of understanding how different emotions are manifested in physiological signals both within and across individuals.

2.5 Computational Novelty of the Work

We introduce a framework for systematic network analysis of human physiological responses. We perform a comprehensive analysis of the effect of different network construction and clustering parameters, concluding that the choice of each parameter can affect the results. For network inference and clustering communities, this highlights the importance of considering various strategies. Nonetheless, we show that via network analysis we can construct image partitions that are statistically significantly similar to the “ground truth” partition, while at the same time they improve it. Importantly, we show that such a result cannot be obtained via a non-network-based analysis of the same data. Thus, viewing affective physiological data through a network lens can improve analysis of the data.

2.6 My Contributions to the Work

I helped design the study and methods, implemented most of the methods, carried out most of the computational analyses, helped analyze the results, and helped write the manuscript.
CHAPTER 3

STUDYING THE INTERPLAY BETWEEN INDIVIDUALS’ EVOLVING
INTERACTION PATTERNS AND TRAITS IN DYNAMIC SOCIAL
NETWORKS VIA LOCAL NETWORK ANALYSIS

The work presented in this chapter has been published as follows:


3.1 Introduction

3.1.1 Motivation

Over the past decade, social network research has experienced a virtual renaissance as considerable repositories of social interactions have been cultivated and explored [113, 159, 201]. The trend has been further accelerated by the rise of social media and the nearly ubiquitous linkage afforded by smart devices and always-on network connectivity. As a result, new questions have emerged about the interplay of social networks and the traits of participants in said social networks. Namely, *to what extent does an individual shape the social networks the individual participates in and/or to what extent do the social networks shape the individual?*

One particular approach is to study the interplay between the social interactions of individuals and the (dis)similarities of their respective traits, such as gender, race, age, religion, and others [126, 192]. Taken further, one can study complex social
determinants on health outcomes including how behaviors such as suicide \cite{17}, obesity \cite{34}, smoking \cite{35}, happiness \cite{62}, and others \cite{63, 68, 152, 192} spread in the social networks. Alternative views explore how not only the traits but also the position of the individual in the social network can affect the development of their own traits \cite{192}. Further applications of linking nodes’ network positions to their traits (i.e., node labels) have yielded considerable insights in other domains as well, such as computational biology \cite{54, 55, 84, 85}.

The natural consequence of these efforts is that coupled with the evidence that social networks are intertwined with traits, it would be highly desirable to distinguish whether the evidence is a consequence of influence, according to which traits are shared between individuals after ties (i.e., edges or links) are formed, or homophily, according to which individuals form (break) ties because they have similar (dissimilar) traits \cite{37, 41, 153, 192, 195, 225}. The challenge is that such a distinction is exceptionally difficult to achieve absent the proper modeling and analysis of individual interactions and traits over time observed over multiple networks. Thus, in this study we introduce a comprehensive framework capable of handling such longitudinal (dynamic) and heterogeneous (multiplex) network data with trait annotations. We note that the goal of answering the foundational question of influence versus homophily goes well beyond the scope of the study. While our framework creates advances towards tackling this question in the long run, its direct short-term goal is to allow for the study of relationships between individuals’ evolving social network positions (while accounting for multiple link types) and their traits.

3.1.2 Related Work

Current related research can be divided along three dimensions: 1) static versus dynamic, 2) homogeneous versus heterogeneous, and 3) global versus local network analysis. We define the three dimensions and then classify the related work accord-
Definitions. A network can represent a static snapshot of the system of interest or can model the system dynamically. In the former case, all nodes and edges are aggregated together and temporal information is ignored (intentionally for simplicity or unintentionally due to limitations of technologies for data collection), which consequently leads to loss of valuable information. In the latter case, the arrival of each node and edge is explicitly recorded [21, 69, 105, 111, 128].

Networks can also be treated as homogeneous or heterogeneous. In the former case, a single link type is studied, or multiple link types are treated equally and aggregated [56, 105, 111, 147, 150]. Critically, as multiple interaction types usually exist (possibly with complementary functionalities), valuable information is lost by simply aggregating the different data types together. In the latter case, different link types are explicitly distinguished, and often, their interplay is studied [42, 46, 128, 200, 223].

Network analysis can further be done on global or local scale. In the former case, one measures global network-level properties, such as the degree distribution [105].
However, global properties might not be able to capture effectively complex topologies of real-world networks, which are overall noisy but often have well-studied local neighborhoods [131]. Thus, studying a network locally from the perspective of nodes (individuals) might be more appropriate, via e.g., centrality measures such as degree, graphlet degree, or betweenness [135, 147, 195]. We note that even though local network analysis focuses on computing node-level properties, global network-wide information might still be used to characterize the local network position of a node. For example, betweenness centrality of a node accounts for shortest paths between all pairs of nodes in the network. Also, we note that there exists a category of network analysis methods that focus on mesoscopic community-level scale, where this scale lies in between the global network-level and local node-level scales [143]. However, this method category is beyond the scope of our study.

**Classification.** Due to limitations of techniques for data collection, social network research has focused on studying static representations of real-world networks and studying networks with a homogeneous link type [3, 17, 53, 64, 68, 139, 147, 150]. With recent advances in data collection techniques, advances have also been made along these two dimensions. Namely, static but now heterogeneous social network analysis has received attention [42, 46, 152, 193, 195, 201, 202] as well as dynamic but homogeneous social network analysis [7, 21, 34, 35, 37, 52, 63, 104, 105, 111, 153]. Similarly, initial attempts have been made in network analysis that are both dynamic and heterogeneous [96]. For example, the evolution of longitudinal social network data encompassing multiple link types [128, 143, 203, 230], as well as the interplay between the different link types [128, 202], have been studied.

To our knowledge, simultaneous network analyses of both dynamic and heterogeneous data have typically been done only from the global (or possibly mesoscopic) but not local scale. Few of the existing studies that have studied local network structure have typically not aimed to link *evolving* multiplex social network structures with
individuals’ traits. For example, whereas studying relationships between local social network structures and individuals’ gender [201] or personality [195] received attention, these studies aggregated all social interactions into a single static network, thus ignoring the valuable temporal information. Another relevant study did compute individuals’ local network properties (i.e., centralities) at different times [152]. However, this study analyzed relationships between the individuals’ personality and their centrality scores averaged over all time points. Clearly, the averaging procedure still discards valuable temporal information, as two nodes with completely different evolving centrality profiles (e.g., increasing versus decreasing centrality with time) could still result in the same average centrality. Although several efforts in the computational biology domain have indeed studied changes in local structure from dynamic molecular networks [51, 85], such networks still encompassed only a single edge type. Thus, efficient analysis of social networks that are both dynamic and heterogeneous remains challenging. With the newfound availability of such complex data, novelties are required in computational frameworks that can cope with foundational network questions.

3.1.3 Our Approach and Contributions

We develop such a framework for: 1) constructing evolving social networks from multiple person-to-person communication types, 2) measuring changes in peoples’ social network patterns with time, 3) clustering people with similar evolving patterns, and 4) linking the clusters with similarities in peoples’ traits (Figure 3.2). We validate our framework on an 18-month dataset encompassing six evolving network types and eleven individual traits. Moreover, we apply our framework to an alternative 9-month dataset encompassing three evolving network types and five individual traits [2]. Importantly, we apply our framework and validate its results open-minded, letting the data reveal to us when there exists a link between the given trait and individuals’
Construct evolving networks encompassing multiple interaction types

Measure changes in node network positions via seven centralities

Cluster nodes based on similarities of evolving network positions

Compare clusters against node traits: do nodes in same cluster share traits?

Figure 3.2: Summary of our work. For details, see Section 3.2.

Our contributions are as follows:

- Unlike most of the related studies (Figure 3.1), we study a social network that is both dynamic and heterogeneous. Also, our link types encompass both digital communications and co-location instances. We note that the level of heterogeneous analysis performed in this work is at analyzing each link type individually and then integrating (i.e., contrasting) results obtained by the different interaction types. Integrating the different link types prior to producing the results is a more advanced step that is subject of our future work.

- Unlike many related studies that focus on both dynamic and heterogeneous network analysis on global scale, we focus on local scale. We measure changes in local network structure via seven node centralities, with several of the centralities drawn from the state-of-the-art research in other domains, e.g., computational biology [54]. By adopting the local properties to social networks, we are further bridging the domains, just as has already been done on the global scale (by e.g., recognizing the scale-free or power-law degree distributions of networks from multiple domains) or on the mesoscopic scale (by e.g., recognizing the property of modularity, i.e., presence of communities, in networks from multiple domains).

- Unlike most of the related studies that focus on both dynamic and heterogeneous social network analysis, we aim to link changes in people’s social network positions to their traits. To do so, we develop a strategy for combining the notions of evolving node centrality and clustering.

- We detect large amounts of network-trait interplay. We validate our results by demonstrating their statistical significance and by showing that the interplay significantly overlaps with ground truth knowledge as well as with the results of existing studies. Notably, we capture unexpected results that might be missed by a simpler approach, such as static network analysis. Also, we uncover a
number of network-trait interplays that have not been studied to date but could lead to interesting insights.

- We present a general framework for studying the interplay between nodes’ network positions and traits. As a proof of concept, we apply the framework to (multiple) social networks. However, it could be potentially applied to numerous domains. Further, since we demonstrate the existence of such interplay in our data and since in many real-world applications it is easier to measure interaction data than nodes’ traits, our methodology can be used to predict traits of uncharacterized nodes based on their network similarities to characterized nodes. Also, in a long run, our framework could be extended to allow for addressing the influence versus homophily question.

3.2 Methods

3.2.1 Dataset

**NetSense dataset.** Our main data is drawn from the NetSense smartphone study at the University of Notre Dame launched in August of 2011. This study monitored the smartphone usage of two hundred freshmen entering the university. Each student was provided with an Android smartphone with a plan giving unlimited data, unlimited texting, and unlimited mobile-to-mobile minutes in exchange for full monitoring privileges of the phone, i.e., of all instances and contexts of communications but not actual message contents. All data gathering was fully approved by the cognizant University of Notre Dame Institutional Review Board (IRB) with full consent forms signed by each participant in the study. All participants were made fully aware of all data gathering for the purposes of the study. For details, see [197].

**Multiplex communication types.** We study digital communications (text messages, phone calls, emails, and Facebook postings) and co-location instances (proximity observed via Bluetooth that captures the potential for face-to-face communications). Each communication event is associated with the people involved in the communication, a timestamp, and a communication length. Each co-location event contains the aforementioned parties involved, timestamp, length of the interaction,
and identification of the relative distance between the two smartphones gleaned via observed signal strength. To filter random encounters for a co-location event, one person needs to detect another continuously for at least six minutes (at least once in the first three minutes and at least once in the last three minutes). The timestamp associated with the event is then the first timestamp of the encounter. Per our guidelines in [118], we divide co-location (i.e., proximity) into close proximity ($\leq 2.5m$, $\geq -55$dbm) and near proximity ($\leq 5.5m$, $\geq -65$dbm). In total, we extract six communication types from the dataset: SMS (S), PhoneCall (P), Email (M), Facebook (F), close proximity (CP), and near proximity (NP). Our data spans 18 months (starting September 2011) and is filtered to 150 users who were involved in the study for the entire period.

**Individual traits.** Five surveys were conducted at different times during the 18-month period. The surveys asked information about participants such as general demographics, prior education, personality, emotional state, and political viewpoints. We extract eleven traits from the surveys describing either a person’s physical conditions (gender, weight, height, health, and BMI) or personality (happiness, agreeableness, extroversion, openness, conscientiousness, and neuroticism; see Table 3.1). We use the trait data to examine a potential link between an individual’s social network position and their traits. The value of each trait is measured as the average of five trait values corresponding to the five surveys, since most of the traits are stable across time, such as gender or height; the only exceptions are happiness and health.

**Friends and family dataset.** In order to test the robustness of our approach to the choice of dataset, in Section 3.3.3 we also study an alternative publicly available social network dataset [2]. This alternative dataset captures communications between 53 subjects from a married graduate student residency of a university over a 9-month period. It contains three communication types: SMS (S), PhoneCall (P),
and Bluetooth (B), and the Big Five personality traits. Thus, this dataset is a strict subset (in terms of communication types) of the NetSense dataset when considering Bluetooth type to be equivalent to close proximity type or near proximity type. For consistency, we construct networks in the same way as for the NetSense dataset (Section 3.2.2). All results except those in Section 3.3.3 are reported for the NetSense dataset.

While both of the considered datasets are small in terms of the number of nodes, our computational framework is applicable to large data as well. However, an extensive search revealed no publicly available large-scale social network that is dynamic, heterogeneous, and annotated with trait information.

3.2.2 Network Construction

To model the dynamic nature of the data, we take data snapshots at different time points and for different communication types (Figure 3.2), with all snapshots covering time intervals of equal length $\Delta t$. For each snapshot, we form a network as follows: nodes are the users (smartphones) and there is a link between two nodes if there are

<table>
<thead>
<tr>
<th>Trait</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreeableness</td>
<td>Affable, tolerant, sensitive, trusting</td>
</tr>
<tr>
<td>Conscientiousness</td>
<td>Organized, systematic, punctual</td>
</tr>
<tr>
<td>Extraversion</td>
<td>Outgoing, talkative, sociable</td>
</tr>
<tr>
<td>Neuroticism</td>
<td>Anxious, irritable, temperamental</td>
</tr>
<tr>
<td>Openness</td>
<td>Curious, intellectual, creative</td>
</tr>
</tbody>
</table>
at least $w$ events of a certain communication type between the two nodes within the given time interval. Network construction parameters $\Delta t$ and $w$ need to be selected carefully such that the resulting networks are meaningful representations of the data. The choice of a small $\Delta t$ or a large $w$ could lead to largely disconnected networks, while the choice of a large $\Delta t$ or a small $w$ could lead to loss of valuable longitudinal information from the data during the given time period. We have previously explored on the same dataset the effects of the choice of $\Delta t$ and $w$ values during the network construction process. Thus, we use the most appropriate values according to the previous work: $\Delta t = 1$ month and $w = 3$. For network statistics of the NetSense and the friends and family datasets, see Figure 3.3 and Appendix Figure B.1 respectively. Importantly, the overall topological trends are similar between the two datasets.

3.2.3 Measures of Network Topology

We measure the social network position of a node in each snapshot via seven centrality measures, each of which aims to capture the topological importance of the
node, often from a complementary perspective compared to other centralities. Hence
the need to use multiple centralities in order to fully capture network characteristics.
The seven measures are: degree centrality (DEGC), clustering coefficient centrality
(CLUSC), k-core centrality (KC), graphlet degree centrality (GDC), betweenness cen-
trality (BETWC), closeness centrality (CLOSEC), and eccentricity centrality (ECC)
[54, 135, 219]. DEGC of a node is the number of its network neighbors while CLUSC
measures the fraction of a node’s neighbors that are also neighbors of each other.
KC of a node is k if the node is in the network’s k-core, where k-core is the maximal
subnetwork in which each node has degree of at least k. GDC measures how many
graphlets (induced 2-5-node subgraphs) a node touches, thus capturing the density
and complexity of the node’s extended network neighborhood in a mathematically
rigorous manner. BETWC of a node is the percentage of shortest paths in the net-
work that go through the node while CLOSEC of a node measures how close (in
terms of the shortest path distance) the node is to all other nodes. ECC of a node
measures how close the node is to the furthest (i.e., most distant) of all nodes.

3.2.4 Network-based Evolution Clustering

We partition all nodes into non-overlapping clusters based on pairwise “similar-
ities” of their evolving social network positions, i.e., centrality values at different
network snapshots of the given communication type. Formally, we define the sim-
ilarity of two nodes a and b as follows. For a given network type and centrality
measure, let $c^i_a$ and $c^i_b$ be the centrality values of a and b, respectively, at time $i$.
Let vectors $V_a$ and $V_b$, denoted as $(c^1_a, c^2_a, \ldots, c^n_a)$ and $(c^1_b, c^2_b, \ldots, c^n_b)$, respectively,
be evolving centrality profiles of the two nodes from time 1 to time n. Then, we
compute the similarity between a and b by calculating 1) Pearson correlation and 2)
Euclidean distance between $V_a$ and $V_b$. Instead of using Pearson correlation which
results in random-like accuracy (Appendix Table B.1) according to our evaluation
strategy (Section 3.2.5 and 3.3.2), we only consider Euclidean distance. We partition
the nodes via: 1) \textit{k-medoids} clustering and 2) \textit{hierarchical} clustering (with single
linkage). Both of the clustering methods require the number of desired clusters, \( k \),
as input. We vary \( k \) from 2 to 50 and evaluate the partition for each \( k \) via popular
partition quality tests: \textit{Dunn index} \cite{Dunn1973}, \textit{silhouette} \cite{Rousseeuw1987},
\textit{connectivity} \cite{Rousseeuw1987}, \textit{the sum of squared error} \cite{Xu1993}, and \textit{Wilcoxon rank-sum test} \cite{Wilcoxon1945}. Overall, the best \( k \) suggested
by each of the above partition quality test falls between 10 to 20 for \textit{k-medoids}, while
hierarchical clustering produces random-like partitions regardless of the value of \( k \).
That is, for hierarchical clustering, according to the sum of squared error method,
intra-cluster similarities in network-based partitions are very close to intra-cluster
similarities in random partitions. Because of such random-like behavior of hierarchi-
cal clustering, in our study, we only consider \textit{k-medoids} clustering. Then, we consider
all even values of \( k \) in the above suggested 10-20 range.

3.2.5 Evaluating Partition Quality

To evaluate a partition of nodes into clusters with similar evolving social network
patterns, we compare this partition with trait-based partitions. That is, for each node
trait, we generate a trait-based partition by clustering the nodes into eight groups
using \textit{k-medoids} method. (We note that we tested other parameter values for the
number of clusters as well, including four and 10, but the value of eight resulted in
the highest accuracy in terms of the evaluation strategy described in Section 3.3.2.)
The only exception is the gender-based partition, which is naturally divided into two
clusters: female and male.

To test the robustness of our approach to the choice of strategy for creating a
trait-based partition, in Section 3.3.3 we also adopt an alternative \textit{equal size} trait-
based partitioning strategy \cite{McMinn2015}: for each node trait, mimicking existing work \cite{Caldarelli2002},
we divide the nodes into eight clusters of equal size, where the clusters are created as follows. If for a given trait all nodes are sorted in increasing order of their trait values, the first cluster contains initial 12.5% of the sorted nodes, the second cluster contains next 12.5% of the sorted nodes, and so on. All results except those in Section 3.3.3 are for the main $k$-medoids trait partitioning strategy.

Given a network-based partition and a trait-based partition, we use *adjusted for chance information distance (AID)* \cite{82, 83} to determine the similarity of the two par-
tions from an information-theoretic perspective. AID quantifies how much knowing one partition reduces uncertainty about knowing the other. The lower the AID value, the more similar the two partitions. AID adjusts for the two compared partitions potentially having different cluster sizes. Intuitively, AID allows us to quantify whether a network-based partition and a trait-based partition are significantly similar, i.e., whether evolving network structure indeed encodes trait information, as we hypothesize. Further, AID allows us to determine which of all network-based partitions (corresponding to different communication types or centrality measures) is the most similar to a trait-based partition. We compute the statistical significance of an AID score empirically [82]. We use an AID $p$-value threshold of 0.01.

Also, we evaluate the quality of a network-based partition with respect to trait similarity via *Wilcoxon rank-sum test* (WRST). Now, a network-based partition is of good quality with respect to trait similarity if it results in significantly higher (lower) intra-cluster than inter-cluster trait similarity (distance); we use a WRST $p$-value threshold of 0.01. For a given trait, we compute the trait distance between two nodes as the absolute value of the difference between their trait values.

For each of AID and WRST, and for each combination of network type, centrality measure, clustering distance (i.e., similarity) metric, and trait, we evaluate the quality of a network-based partition with respect to trait similarity for each of the $k$ values (i.e., for all even values in the optimal 10-20 range; Section 2.4), and we report results only for the best of these values, i.e., for the value of $k$ with the lowest $p$-value.

3.3 Results and Discussion

3.3.1 Detection of Network-Trait Interplay

For each of six network communication types (Section 3.2.1) and for each of seven node centralities (Section 3.2.3), we generate a network-based partition by clustering
Figure 3.4: Interplays between a network type (see the given matrix column) and a trait (see the given matrix row) detected in our NetSense dataset via at least one centrality by AID-ED and by: (a) our dynamic network analysis, or (b) dynamic network analysis only (black), static network analysis only (striped), both dynamic and static network analysis (grey), or neither dynamic nor static network analysis (white).

together nodes whose network positions (i.e., centrality values) evolve similarly (Section 3.2.4). Then, we aim to test whether nodes in the same network-based clusters share similar traits. We do so by comparing a network-based partition with a trait-based partition (Section 3.2.5), for each of the traits (Section 3.2.1). We say that we detect the interplay between a network type and a trait with respect to a given centrality measure if the network-based partition and the trait-based partition are statistically significantly similar ($p$-value below 0.01) with respect to AID combined with ED (Section 3.2.5). As we will show below, other approaches (WRST combined with ED, WRST combined with PC, and AID combined with PC) do not seem to detect meaningful interplays. Thus, henceforth we only discuss the interplays detected by AID combined with ED (Table 3.2).

For each network type, there are $7 \times 11 = 77$ different combinations of node centralities and node traits. We detect the interplay for at least one network type for 14 of these 77 combinations (Table 3.2). Because different centrality measures
often capture complementary aspects of the network position of a node, results of the different centrality measures are typically combined together [54]. When we do this, we detect the interplay for at least one network type for nine of the 11 traits, and we detect the interplay for at least one trait for six of the network types. That is, we detect the interplay between almost each trait and at least one network type, which we interpret qualitatively below, after we validate our results below.

Several more detailed quantitative observations of our results from Table 3.2 are as follows. In terms of network types, Facebook (F) is the best in the sense that it works (i.e., detects the interplay) for the most (four) of the 77 combinations of node centralities and node traits (see above). Facebook (F) captures two node traits: gender (with respect to three centralities) and neuroticism (with respect to one centrality). In terms of node centralities, CLOSEC works the best, as it captures the most (four) traits, sometimes with multiple network types per trait. DEGC and BETWC on the other hand perform poorly, as each captures only one interplay. It is encouraging that GDC is superior to DEGC, as GDC is a sensitive measure of network topology [135], while DEGC captures only the direct network neighborhood of a node and thus limited network topology.

Clearly, different centralities do not always show consistent results in Table 3.2. As we have already argued above, this is not surprising, as different centralities have been designed intentionally to (partly) complement each other [54]. Thus, as is commonly done [54], henceforth, we say that we detect the interplay between a network type and a trait if at least one of the centrality measures supports the interplay (Figure 3.4(a)). With this approach, of all traits, gender is captured by the most (three) of six network-based partitions (corresponding to the six network types). Happiness and health on the other hand are captured by the fewest (none) of the six network types. Of all network types, PhoneCall (P) captures the most (three) of the 11 traits. Mail (M) and SMS (S) on the other hand capture the fewest (one)
of the traits. Each network type is able to capture at least one node trait.

In summary, we detect large amounts of the interplay between evolving network structure and traits. After we validate the above results in Section 3.3.2, in Section 3.3.4 we relate them to results from the existing literature on communications and sociology.

3.3.2 Validation of Our Results

3.3.2.1 (Dis)similarities between Traits

We aim to validate the detected interplays by measuring how well they capture expected “ground truth” (dis)similarities between different traits. First, we determine which pairs of traits are expected to be significantly similar or dissimilar, i.e., which trait pairs are correlated or uncorrelated according to the ground truth trait data.
Then, we aim to validate our results by measuring the percentage of the correlated (positive control) or uncorrelated (negative control) ground truth trait pairs that are also similar in terms of their interplay with the evolving network structure. A high percentage for positive control trait pairs and a low percentage for negative control trait pairs would validate our results.

To measure which trait pairs are (un)correlated according to the ground truth trait data, we compare two trait-based partitions corresponding to two given traits, and we do this in the same way as when we compare a network-based partition with a trait-based partition – by using AID or WRST (Section 3.2.5). With this strategy, there are six and 15 correlated (i.e., significantly similar) trait pairs according to AID and WRST, respectively, with all six pairs in the overlap (Figure 3.5). That is, all correlated pairs detected by AID are also detected by WRST and are thus more credible than those correlated pairs detected only by WRST. Thus, henceforth, we only focus on correlated pairs detected by AID (which are also supported by WRST).

It is encouraging that gender, height, BMI, and weight are all highly interconnected (Figure 3.5), as these four traits are all related to a person’s physical appearance and they typically relate to each other in real life. Further, it is encouraging that the link is captured between agreeableness and happiness, which is consistent to existing findings in the literature [28]. On the other hand, the Big Five personality traits (i.e., agreeableness, extroversion, openness, conscientiousness, and neuroticism) are mostly disconnected by our approach. This is also encouraging, as the Big Five personality traits account for five broad domains in personality and are thus relatively independent [16]. Even when we identify with our approach the (dis)similarities among the Big Five personality traits based on the trait data of the *friends and family* dataset (which are needed for interpretation of our results below), most of these traits are also disconnected, which further validates our approach (Appendix Figure B.2).
Thus, since we demonstrate that our positive and negative controls (Figure 3.5) make sense, we rely on these controls to validate our detected interplays above. Specifically, we measure the “validation rate” for both correlated positive control trait pairs and uncorrelated negative control trait pairs, where by validation rate we mean the fraction of the control trait pairs that share at least one network type in Figure 3.4(a) i.e., trait pairs that are also similar in terms of their interplay with evolving network structure. If our detected interplays above are meaningful, we should observe a high validation rate for positive control pairs and a low validation rate for negative control pairs. Indeed, this is what we observe. The interplays (detected by AID when combined with ED) capture 43% of the correlated trait pairs but only 5% of the uncorrelated pairs; thus, validation rate of the positive controls improves by 714% over validation rate of the negative controls (Table 3.3). We note that similar does not hold for the interplays detected by AID when combined with PC, nor for the interplays detected by WRST when combined with any of ED or PC, as in these cases, the two validation rates are similar (Appendix Table B.1 and
This is exactly why in our study we have focused only on the interplays detected by AID when combined with ED. Understanding why the other parameter values do not lead to meaningful results is beyond the scope of this study.

3.3.2.2 Statistical Significance of Our Results

To ensure that our above findings cannot be detected by chance, we repeat the entire dynamic analysis (as we do in Sections 3.3.1 and 3.3.2.1) on a randomized dataset (instead of the actual real dataset) that is generated by one of two randomization approaches (see below). Our dynamic network analysis framework should not lead to meaningful results from the randomized data. In other words, the framework’s improvement of positive over negative control validation rate should be significantly lower for the randomized data than for the actual real data.

We design the first randomization approach, which we refer to as “randomized centralities,” as follows. For a given network type and centrality measure, and for each node, we randomly shuffle the node’s centrality values across different network snapshots. By randomizing the data in this way, the average centrality value of each node over different times remains the same, but the evolving centrality pattern of the node is randomized. We design the second randomization approach, which we refer to as “randomized clusters,” as follows. For a given network type and centrality value, and for the corresponding partition $P$ that clusters nodes with similar evolving centrality patterns, we produce a randomized counterpart $P'$ of the partition $P$, where $P$ and $P'$ have the same number of clusters and the same distribution of cluster sizes, but the nodes are assigned to clusters in $P'$ randomly, whereas they are assigned to clusters in $P$ based on the nodes’ evolving centrality patterns (as measured from the actual real data). For each of the two randomization approaches, we run our dynamic analysis on the randomized data 30 times; we are limited to this number of runs due to high computational complexity. Then, we report the average and standard deviation...
TABLE 3.3

VALIDATION RATES OF OUR RESULTS ON THE NetSense DATASET
(FOR AID-ED COMBINATION), IN TERMS OF EXPECTED
TRAIT-TRAIT (DIS)SIMILARITIES, FOR OUR PROPOSED
DYNAMIC NETWORK ANALYSIS FRAMEWORK, ITS TWO
RANDOMIZED COUNTERPARTS, AND ITS STATIC COUNTERPART

<table>
<thead>
<tr>
<th>Analysis type</th>
<th>Positive control validation rate</th>
<th>Negative control validation rate</th>
<th>Improvement of positive over negative controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic analysis</td>
<td>43%</td>
<td>5%</td>
<td>714%</td>
</tr>
<tr>
<td>Random dynamic analysis (randomized centralities)</td>
<td>20% (19%)</td>
<td>17% (10%)</td>
<td>39% (153%)</td>
</tr>
<tr>
<td>Random dynamic analysis (randomized clusters)</td>
<td>54% (21%)</td>
<td>50% (15%)</td>
<td>14% (55%)</td>
</tr>
<tr>
<td>Static analysis</td>
<td>29%</td>
<td>16%</td>
<td>81%</td>
</tr>
</tbody>
</table>

* For the randomized approaches, the numbers outside and inside of the parentheses are the averages and the corresponding standard deviations, respectively. The improvement is measured as the difference between positive control validation rate and negative control validation rate, divided by negative control validation rate. Note that all reported percentages are rounded.

Indeed, as we have hypothesized, our dynamic network analysis framework results in a significantly higher improvement \( p < 10^{-5} \) via the Z-test of positive over negative control validation rate when run on the actual real dataset than on each of the randomized datasets (Table 3.3). This confirms meaningfulness of our results from Sections 3.3.1 and 3.3.2.1.
3.3.3 Robustness of Our Approach

We evaluate the robustness of our approach to the choice of trait partitioning strategy. Recall that when we study the NetSense dataset (Section 3.2.1) using the main $k$-medoids trait-based partitioning strategy (Section 3.2.5), we detect 11 network type-trait interplays. When we do the same except that we use the alternative equal size trait-based partitioning strategy (Section 3.2.5), we detect 13 interplays. The majority ($9/11 = 82\%$) of the interplays are in the overlap of the two trait-based partitioning strategies (Figure 3.7(a)), and this overlap is statistically significant ($p$-value of 0.02 via the hypergeometric test). This indicates robustness of our approach to the choice of trait-based partitioning strategy. This is confirmed even further when we study the friends and family dataset (Section 3.2.1 and Figure 3.7(b)): again, the majority ($5/7 = 71\%$) of the interplays are in the overlap. For further results regarding the choice of trait-based partitioning strategy, see Appendix Figures 3.3-3.5.

Also, we evaluate the robustness of our approach to the choice of dataset. Here, we focus only on the Big Five personality traits, as these five traits are common to our NetSense data and the friends and family data. In this context, when we use the main $k$-medoids partitioning strategy to analyze the NetSense dataset, we detect four network type-trait interplays. When we do the same except that we use the friends and family dataset, we detect seven interplays. Half ($2/4 = 50\%$) of the interplays are in the overlap between the two datasets (Figure 3.7(c)). Similar holds when we use the equal size trait-based partitioning strategy instead of the main $k$-medoids strategy (Figure 3.7(d)). Thus, because the majority of the interplays detected from our NetSense data are also detected by the friends and family data, but not the other way around, the former may be more credible than the latter. This is even further confirmed when we compare the two datasets’ improvements of positive over negative control validation rate. This improvement is 200% for our NetSense data (Appendix Table B.3) while it is -13% for the friends and family data (Appendix Table B.3).
Figure 3.7: Interplays between a network type and a trait detected in (a) our NetSense dataset and (b) the friends and family dataset, based on our main k-medoids trait-based partitioning strategy only (black), the alternative equal size trait-based partitioning strategy only (striped), both strategies (grey), or none of the strategies (white), and interplays detected based on (c) our main k-medoids trait-based partitioning strategy and (d) the alternative equal size trait-based partitioning strategy in our NetSense dataset only (black), the friends and family dataset only (striped), both datasets (grey), or neither of the datasets (white). In panels (b)-(d), we show only network types and traits that are common to the two datasets, to allow for a fair result comparison.

All of these results suggest that our approach produce better predictions from the NetSense dataset than from the friends and family dataset. This could be due to the limited size of the friends and family dataset (Section 3.2.1).

3.3.4 Comparison of Our Results with Findings from the Literature

To interpret implications of our results (from the NetSense dataset when using the main k-medoids trait-based partitioning strategy), we next compare our interplay findings versus findings from the literature of existing studies that have also aimed to link cell phone use with individual traits.
We detect interplay between PhoneCall and agreeableness. This is encouraging, as follows. Disagreeable people were reported to participate in more incoming phone calls than agreeable people [26]. Agreeableness is a trait that is linked to interpersonal relationships, where such relationships are based on equal and honest exchange of information. So, the large number of incoming calls for disagreeable people may be explained by other people being inclined to communicate with a disagreeable person via phone rather than to interact with them face-to-face [26]. We also detect interplay between PhoneCall and extraversion. This link is supported by literature evidence as well. It was argued that extraverts are more likely to receive calls than introverts, where the total duration of their calls and the number of their unique callers are likely to be higher as well [33]. Further, the number of unique contacts associated with voice calls is increased for non-conscientious users compared to conscientious users [33], which is encouraging, since we detect interplay between PhoneCall and conscientiousness as well. Moreover, we detect interplay between SMS and openness, and a literature evidence showed that the number of SMSs sent or received is negatively correlated to openness [33]. Thus, it is encouraging that many of the interplays detected by our approach on the NetSense dataset are supported by the existing body of work. Further, our approach captures some additional interplays from the friends and family dataset, such as between SMS and neuroticism, or between Bluetooth (B) and agreeableness. These findings are supported in the literature as well. For example, neuroticism could explain time spent SMSing, and people who are more socially anxious and lonely prefer to use SMS due to feeling that they can better express themselves [26]. Also, the number of Bluetooth IDs seen for long duration of time is likely to be higher for disagreeable users [33].

An existing study that is very relevant for our work in terms of both methodology and goals aimed (among other things) to measure which of two different communication types (phone call denoted by P in our study) and Bluetooth proximity (denoted
by CP or NP in our study) relates better to the big-five personality traits [195]. It did so by classifying (via supervised learning) the nodes in terms of their network-based features (centrality values) in a static social network, and by measuring how accurately the classifier predicts the node traits. (We ask the same question but for five communication types, with clustering via unsupervised learning, and from a dynamic network.) In addition, this existing work [195] aimed to do the same by classifying nodes in terms of their actor-based features (such as the number of outgoing and incoming calls, maximum time an individual was seen in proximity by the actor, etc.) rather than in terms of the node centralities.

Findings of this study [195] were as follows. When using the network-based features, the proximity network outperformed the phone call network most of the time in terms of accuracy of predicting the five personalities. However, when using the actor-based features, the phone call network outperformed the proximity network most of the time. Interestingly, the results obtained with respect to the network-based features were the opposite of the results obtained with respect to the actor-based features. It is encouraging that our findings mostly agree with the existing actor feature-based results [195], even though these two result sets disagree with the existing network feature-based results [195].

A potential explanation for this observed disagreement could be that our results have been obtained from a dynamic network while the existing network feature-based results have been obtained from a static network. Thus, static network analysis could be performing sub-optimally (as we will demonstrate in Section 3.3.5), which only emphasizes the advantage of our dynamic approach. Interestingly, even when we compare these existing results obtained from the static data [195] with our results obtained from a static counterpart of our dynamic framework (Section 3.3.5), there is a mismatch. This is unlikely due to the choice of the dataset. Namely, this existing study [195] also uses the friends and family dataset that we analyze in our work.
And our results for the friends and family dataset are still consistent with our results for the NetSense data (Section 3.3.3), while both are inconsistent to the existing study's results for (their version of) the friends and family dataset. We note that the publicly available friends and family dataset that we use from [2] (Section 3.2.1) and that originates from the same authors as the existing study [195] is slightly different than the dataset used in the existing study itself [195]; a key difference between the two versions of the friends and family dataset is in terms of the duration (nine months in our study versus three months in [195]).

An alternative reason behind the above observed disagreement (our results not matching the existing network feature-based results [195]) could be that the proximity communication data from the existing study might be noisy, as no threshold was used to filter out smartphones detected via weak signals [195], while we do filter out such devices with the goal of avoiding noise in the data (which was a particular focus of our prior work [118]) and also we do filter out random encounters that are likely a result of co-location (for example, one person needs to detect another continuously for at least six minutes; Section 3.2.1). Clearly, results obtained from any subsequent analysis of potentially noisy proximity data are likely to be noisy themselves.

Another relevant study in terms of both methodology and goals aimed to incorporate some network dynamics when learning the relationship between individuals’ network-based features (degree and betweenness) and their personality, where the network-based features were obtained by averaging the nodes’ centrality values across different time points (i.e., network snapshots) [152]. The network-based features were measured on two types of networks, including face-to-face proximity interactions (as measured by sociometric badges’ IR sensors, which can be thought as a proxy of our CP or NP network types) and face-to-bed interactions. In the latter network type, a node is either a bed or a person, and a link exists between a bed and a person if the person is in proximity to the bed. Since the face-to-bed network type contains
more than one node type, this network type is out of the scope of our study, and thus, we do not discuss it further. In addition, this existing study [152] analyzed the relationship between individuals’ actor-based features (i.e., face-to-face time, number of people with face-to-face, and time spent in close proximity to other people) and their personality.

Findings of this study [152] were as follows. When using the network-based features, no correlation was observed between the average centrality values and the personality traits, although there was significant correlation (at our study’s $p$-value threshold of 0.01) between the standard deviation of one centrality measure (betweenness) and two of the five personality traits (openness and conscientiousness). When using the actor-based features, there was no significant correlation between any of these two features and any of the five personality traits. Since in our study we detect no interplay between the proximity network and any of the personality traits, our results are consistent with the results of this existing study [152] when using the actor-based features, and our results are partly consistent (in terms of the centrality averages but not in terms of the standard deviations) with the results of this existing study [152] when using the network-based features.

We note that the reasons why we have focused on evaluating our approach on the friends and family dataset from [195] (i.e., from the same authors’ previous work [2]; see above) rather than on the dataset from [152] are as follows. First, the latter study [152] uses more limited temporal social interaction dataset, which spans only 27 days, while the former study uses social interaction dataset that spans nine months [2] and is thus more applicable to our approach’s parameter choices, such as those related to network construction (Section 3.2.2). Second, the latter [152] is an older study (from 2009) compared to the former (from 2011) [2], and thus, the former is likely of higher quality. Third, while the latter study’s dataset was collected by sociometric badges that were only carried by the users during the work time, both the former and
our study’s datasets were collected by smartphones that were carried by the users all day around; thus, the former [2] is more similar to our study in terms of the data collection technologies.

3.3.5 On Dynamic versus Static Network Analysis

Importantly, even when we repeat our analysis on our data in exactly the same way as done so far but on the aggregated static representation of the data, we get mostly complementary insights from our original dynamic network analysis compared to this static analysis. (We note that the absolute difference between node centrality values is used as the clustering distance metric for static analysis.) Namely, 11 network type-trait interplays have been detected via our dynamic framework as well as via static network analysis, with only three interplays in the overlap (Figure 3.4(b)). To examine which interplays are more meaningful, we compare improvement of positive over negative control validation rate for dynamic analysis with the same improvement for static analysis. The results indicate drastic superiority of our dynamic analysis (Table 3.3). Similar results hold independent of the choice of dataset or trait-based partitioning strategy (Appendix Figures B.3-B.5).

3.4 Conclusions

Our framework validates existing network type-trait relationships. Also, it reveals additional relationships that are missed by the simpler approaches such as static network analysis alone or that have not been studied to date. Further analyses of these relationships by domain experts could lead to novel insights.

Ultimately, our framework could be extended to allow for in-depth analysis of relationships between evolving social network structures and evolving node traits, with the goal of studying the effect of network structure on the spread of traits, as well as the effect of the spread of traits on the formation of network structure.
Studying these effects is important for developing an efficient predictive model of network structure from trait information and vice versa, as well as of one network type (or trait) from another network type (or trait).

3.5 Computational Novelty of the Work

We present a general framework for studying the interplay between nodes’ network positions and traits. Unlike most of the related studies, we study social networks that are both dynamic and heterogeneous. While many of the related studies focus on both dynamic and heterogeneous network analysis on global scale, we focus on local scale. Furthermore, unlike most of the related studies that focus on both dynamic and heterogeneous social network analysis, we aim to link changes in people’s social network positions to their traits. To do so, we develop a strategy for combining the notions of evolving node centrality and clustering. Notably, our framework can capture unexpected results that might be missed by a simpler approach, such as static network analysis. Also, we uncover a number of network-trait interplays that have not been studied to date but could lead to interesting insights.

3.6 My Contributions to the Work

I helped analyze the results and helped write the manuscript.
PART II

NOVEL NETWORK APPROACHES FOR ESTABLISHED RESEARCH QUESTIONS
CHAPTER 4

REVEALING MISSING PARTS OF THE CELLULAR INTERACTOME VIA GRAPHLET-BASED LINK PREDICTION

The work presented in this chapter has been published as follows:


4.1 Introduction

4.1.1 Motivation

Networks (or graphs) model real-world phenomena in many domains. We focus on biological networks, protein-protein interaction (PPI) networks in particular, with the goal of identifying missing and spurious links in current noisy PPI networks. Nonetheless, our study is applicable to other network types as well. In PPI networks, nodes are proteins and two nodes are connected by an edge if the corresponding proteins interact in the cell. We focus on these networks, since it is the proteins (gene products) that carry out the majority of cellular processes and they do so by interacting with other proteins. And this is exactly what PPI networks model.

High-throughput methods for PPI detection, e.g., yeast two-hybrid (Y2H) assays or affinity purification followed by mass spectrometry (AP/MS), have produced PPI data for many species [24, 67, 190, 196, 226]. However, current networks are noisy, with many missing and spurious PPIs, due to limitations of biotechnologies as well as human biases [38, 71, 198, 213, 217]. AP/MS is estimated to have a 15-50% false
positive rate and a 63-77% false negative rate \[50\]. Similar holds for Y2H, though PPIs obtained by Y2H are still more precise than literature-curated PPIs supported by a single publication \[210\].

Analogous to genomic sequence research, biological network research is promising to revolutionarize our biological understanding: prediction of protein function and the role of proteins in disease from PPI network topology has already received attention \[14, 184-186, 208\]. However, the noisiness of the network data is an obstacle on this promising avenue, as it could lead to incorrect predictions. Computational de-noising of current PPI network data by identifying missing and spurious links could improve the quality of topology-based predictions and consequently save resources needed for experimental validation of the predictions. Thus, we aim to test how well we can decrease the noise in PPI data via link prediction (LP).

4.1.2 Related Work

LP typically uses the existing topology of the network to predict missing and spurious links \[66, 114, 115, 119, 173, 180\]. Alternatively, one network type, e.g., functional interactions, can be used to predict another network type, e.g., physical PPIs \[180\]. LP consists of unsupervised or supervised approaches that use some measure of the topology of the nodes to be linked \[115\]. For example, it may be desirable to link nodes with high degrees as measured by preferential attachment \[13, 144, 220, 232\], nodes that share many neighbors as measured by Jaccard \[179\] or Adamic/Adar coefficients \[1\], nodes that share many paths as measured by Katz index \[92\], or similar \[228\]. Both supervised and unsupervised LP methods have their (dis)advantages. Though supervised methods can outperform unsupervised ones, much of previous research has focused on unsupervised LP, since many factors that might influence supervised LP have not been well understood \[114, 115, 119\].

There are some limitations to the existing LP measures. With some exceptions
most of them capture only the topological information contained in the \textit{immediate} network neighborhood of nodes to be linked \cite{1, 13, 144, 179, 233}. However, significant amount of the information is available in the rest of the network that could improve LP accuracy. Thus, additional sensitive measures that capture deeper network topology might be needed. We recently generalized the idea of shared immediate neighborhoods to shared \textit{extended} neighborhoods in the context of network clustering and showed that including more network topology resulted in biologically superior clusters \cite{194}. So, it is reasonable to test whether including more topology will be effective for LP as well.

Also, most of the existing shared neighborhood-based methods can predict a link only between nodes that are within the shortest path \textit{distance of two} from each other \cite{1, 179, 233}, whereas it might be beneficial to link nodes which are \textit{more distant} \cite{92, 110}. Preferential attachment-based measures can achieve this \cite{13, 144}, but they again capture only the immediate neighborhoods of the nodes to be linked. A shortest path-based LP method exists which can also connect distant nodes in the network but which can at the same time capture deeper network topology. However, this method is computationally expensive \cite{101}. A couple of additional methods exist that can link distant nodes under the hypothesis that nodes that share many paths or that are at similar distances to all other nodes in the network should be linked \cite{92, 110}. Here, we introduce an alternative and sensitive measure of the topological similarity of \textit{extended} neighborhoods of two nodes that addresses all of the above issues, and we use it with a \textit{novel hypothesis} that nodes that are topologically similar should be linked together.

Another drawback of the existing methods is as follows. It might be more efficient to predict the existence of a link between two nodes by explicitly measuring the topological position of an \textit{edge} (or equivalently a non-edge) rather than by measuring the position of each of the two \textit{nodes} individually, as the current methods
do [16]. Thus, we propose a new, sensitive measure of the network position of an edge and a non-edge, which counts the number of subgraphs that the two nodes in question participate in simultaneously, and we use it with the hypothesis that nodes that participate in many subgraphs and thus have large and dense extended shared neighborhoods should be linked together.

4.1.3 Our Approach and Contributions

We study several PPI networks of yeast, the best studied species to date, obtained by different experimental methods for PPI detection, and we apply our new as well as popular existing LP measures to the networks to de-noise them. Given a network, we aim to study the topologies of each node pair in it with respect to the given LP measure, in order to determine which of the node pairs should be connected.

We perform three types of evaluation tests, as follows. First, we introduce synthetic noise in the given PPI network by randomly removing a percentage of edges from the network, with the goal of measuring how well the given method can reconstruct the original network, using the original PPIs as the ground truth data. Second, given the availability of low-confidence PPI data for one of the studied networks, we apply the given method to this network and use the corresponding low-confidence PPI data as the ground truth data when evaluating the method. In both of these evaluation tests, we test the accuracy of a LP method in systematic receiver-operator curve and precision-recall settings. In this context, we study the effects on LP accuracy of the “topological similarity” as well as size of the shared extended neighborhoods of nodes, where the nodes can be distant in the network. Also, we study what amount of network topology should be used for LP. We find that LP measures which favor nodes which are both topologically similar and which have large shared extended neighborhoods are superior to LP measures that have only one of these two properties. Also, we show that using more network topology
often though not always increases LP accuracy.

Third, we apply the LP methods to the original PPI networks to de-noise them, and we evaluate the quality of the de-noised networks, i.e., of different LP methods, in two ways. First, we compute their biological correctness by measuring the “enrichment” of predicted edges in Gene Ontology (GO) terms. Importantly, we show that LP improves the biological correctness of the PPI networks by de-noising them. Second, we search for the predicted interactions in an external, independent PPI data source, and in this way, we validate a significantly large portion of the predictions, further confirming the biological correctness of the de-noised networks.

Importantly, we show that our new LP measures are statistically significantly superior to each of the existing ones with respect to at least one of the evaluation criteria. Alarmingly, we find that receiver-operator curve, precision-recall, and biological (functional) evaluation frameworks do not necessarily agree in identifying the best LP method(s), which has important implications for the LP community.

4.2 Methods

We study multiple S. cerevisiae PPI networks obtained by different experimental methods for PPI detection. Given a network, we aim to de-noise the network, with the goal of determining which of all pairs of nodes in the network should be connected by edges, with respect to a variety of existing as well as new LP measures. We evaluate the different measures in systematic precision-recall and receiver-operator curve frameworks, as well as with respect to two biological (functional) criteria. The details are as follows.

4.2.1 Network Data

We evaluate all LP methods on three S. cerevisiae yeast PPI networks obtained with different experimental methods. We study PPI networks of yeast because yeast
has been the most studied species to date. As such, it has the most complete interactome and thus represents the best species to evaluate the methods on. We study multiple yeast PPI networks obtained with different experimental methods for PPI detection to test whether LP results are dependent on the experimental method. The three networks are: 1) Y2H network, obtained by Y2H, which consists of 1,647 nodes and 2,518 edges [194, 226]; 2) AP/MS network, obtained by AP/MS, which consists of 1,004 nodes and 8,319 edges [194, 226]; and 3) high-confidence (HC) network, obtained from multiple data sources, which consists of 1,004 nodes and 8,323 edges [38]. The quality of PPIs in the HC network is comparable to the quality of interactions produced by precise small-scale biological experiments [38]. Importantly, in addition to the high-confidence PPIs, the data by [38] also contains the corresponding lower-confidence PPI data, which is useful for evaluation of the LP methods (as explained below).

4.2.2 Existing Commonly Used LP Measures

4.2.2.1 Degree-based Measure

According to preferential attachment [13, 144], the higher the degrees of two nodes, the more likely the nodes are to interact. The degree product (DP) measure scores the potential edge between two nodes \( v \) and \( w \) as: \( \text{DP}(v, w) = d(v) \times d(w) \), where \( d(v) \) is the degree of node \( v \) [115].

4.2.2.2 Common Neighbors-based Measures

A popular idea is that the more neighbors (or paths) two nodes share, the more likely the nodes are to interact. Hence, a number of methods has been proposed in this context, as follows.

The shared neighbors (SN) measure scores the potential edge between nodes \( v \) and \( w \) as: \( \text{SN}(v, w) = |N(v) \cap N(w)| \), where \( N(v) \) is the set of neighbors of \( v \) [13, 144].
SN simply counts the shared neighbors.

**Jaccard coefficient (JC)** scores the potential edge between two nodes $v$ and $w$ as:

$$JC(v, w) = \frac{|N(v) \cap N(w)|}{|N(v) \cup N(w)|}.$$  

That is, it scores two nodes with respect to the size of their shared neighborhood relative to the size of their entire neighborhoods combined. As such, it favors node pairs for which a high percentage of all neighbors are shared.

The **Adamic-Adar (AA)** measure scores the potential edge between two nodes $v$ and $w$ as:

$$AA(v, w) = \sum_{z \in N(v) \cap N(w)} \frac{1}{\log d(z)}.$$  

Thus, of all common neighbors of two nodes, it favors low-degree shared neighbors over high-degree shared neighbors.

**Katz index (Katz)** scores the potential edge between two nodes $v$ and $w$ as follows:

$$Katz(v, w) = \sum_{l=1}^{\infty} \beta^l |paths_{v,w}^{<l>}|,$$

where $paths_{v,w}^{<l>}$ is the set of all paths between $v$ and $w$ having length of exactly $l$, and $\beta > 0$ is a parameter that controls relative weights (i.e., levels of importance) of paths of different lengths [92], such that the smaller the value of $\beta$, the smaller the contribution of larger paths is to the sum. For our evaluation, just as in the original publication [92], we use $\beta = 0.005$. In summary, Katz favors node pairs that share many paths of different lengths.

**Local path index (LPI)** scores the potential edge between two nodes $v$ and $w$ as follows:

$$LPI(v, w) = \sum_{l=2}^{3} \beta^l |paths_{v,w}^{<l>}|.$$  

By considering paths of length $l = 2$ and $l = 3$, LPI provides a trade-off between SN (which considers only $l = 2$) and Katz (which considers all possible values of $l$).

**Resource allocation index (RAI)** scores the potential edge between two nodes $v$ and $w$ as:

$$RAI(v, w) = \sum_{z \in N(v) \cap N(w)} \frac{1}{d(z)}.$$  

This measure, motivated by the resource allocation process taking place on networks, is similar to AA, the only difference being scaling of the denominator. For networks with small average degree, the results of AA and RAI are expected to be similar [233].

**Random walk with resistance (RWS)** scores the potential edge between two nodes $v$ and $w$ under the intuition that nodes having similar “distances” to all other nodes in the network are likely to interact with each other [110]; here, the distance is defined
as the expected number of steps needed for a random walker to travel between two nodes in question. As such, RWS can predict links between nodes that are not necessarily close to each other and thus might not share any common neighbors. For a formal description, see the original publication [110].

4.2.3 New LP Measures

We already designed sensitive measures of topology that unlike many of the existing measures go beyond capturing only the direct neighborhoods of nodes to be linked. We used them for network alignment [100, 102, 134], clustering [75, 130, 133, 194], and modeling [127, 132], but they have not been used for LP thus far. Thus, we introduce them as new LP measures. Also, we design conceptually new measures. The details are as follows.

4.2.3.1 Existing Sensitive Measures of Topology as New LP Measures

To go beyond capturing only the direct network neighborhood of a node, we previously designed a constraining graphlet-based measure of topology, called node graphlet degree vector (node-GDV), that captures up to 4-deep neighborhood of a node; a graphlet is a small induced subgraph of the network [165]. We designed a measure of topological similarity of such extended neighborhoods of two nodes, called node-GDV-similarity. In this study, we use node-GDV-similarity for LP, with the hypothesis that the more topologically similar two nodes are, the more likely the nodes are to interact. Also, since shared neighbors-based approaches, which are among the best LP measures over the widest range of real-world networks [115], are based on the number of 3-node paths that two nodes in question share, where a 3-node path is just a 3-node graphlet, we generalize these measures by counting the number of all 3-5-node graphlets that the two nodes share. We do this by using a sensitive measure called edge-GDV. The formal description of all of the measures is
as follows.

**Node graphlet degree vector (node-GDV).** We generalized the degree of node \( v \) that counts the number of edges that \( v \) touches (where an edge is the only 2-node graphlet, denoted by \( G_0 \) in Figure 4.1), into *node-GDV* of \( v \) that counts the number of 2-5-node graphlets that \( v \) touches [130]. We need to distinguish between \( v \) touching, for example, a three-node path (\( G_1 \) in Figure 4.1) at an end node or at the middle node, because the end nodes are topologically identical to each other, while the middle node is not. This is because an automorphism (defined below) of \( G_1 \) maps the end nodes to one another and the middle node to itself. Formally, an isomorphism \( f \) from graph \( X \) to graph \( Y \) is a bijection of nodes of \( X \) to nodes of \( Y \) such that \( xy \) is an edge of \( X \) if and only if \( f(x)f(y) \) is an edge of \( Y \). An automorphism is an isomorphism from \( X \) to itself. The automorphisms of \( X \) form the automorphism group, \( \text{Aut}(X) \). If \( x \) is a node of \( X \), then the automorphism node orbit of \( x \) is \( \text{Orb}_n(x) = \{ y \in V(X) | y = f(x) \text{ for some } f \in \text{Aut}(X) \} \), where \( V(X) \) is the set of nodes of \( X \). There are 73 node orbits for 2-5-node graphlets. Hence, node-GDV of \( v \) has 73 elements counting how many node orbits of each type touch \( v \) (\( v \)'s degree is the first element). It captures \( v \)'s up to 4-deep neighborhood and thus a large portion of real networks, as they are small-world [215].

**Node-GDV-similarity.** To compare node-GDVs of two nodes, one could use some existing measure, e.g., Euclidean distance. However, this might be inappropriate, as some orbit counts are not independent. Hence, we designed a new measure, called node-GDV-similarity, as follows [130]. For a node \( u \in G \), \( u_i \) is the \( i^{th} \) element of its node-GDV. The distance between the \( i^{th} \) orbits of nodes \( u \) and \( v \) is \( D_i(u,v) = w_i \times \frac{\log(u_i+1) - \log(v_i+1)}{\log(\max\{u_i,v_i\}+2)} \), where \( w_i \) is the weight of orbit \( i \) that accounts for orbit dependencies [130]. The \( \log \) is used because the \( i^{th} \) elements of two node-GDVs can differ by several orders of magnitude and we did not want the distance between node-GDVs to be dominated by large values. The total distance is \( D(u,v) = \sum_{i=0}^{72} \frac{D_i}{w_i} \).
Figure 4.1: Graphlet positions of a node, an edge, a non-edge, and a node pair. All topological positions ("orbits") in up to 4-node graphlets of a node (top; node shade), an edge (upper middle; solid line), a non-edge (lower middle; broken line), and any node pair, an edge or a non-edge (bottom; wavy line) are shown. For example: 1) in graphlet $G_3$, the two end nodes are in node orbit 4, while the two middle nodes are in node orbit 5; 2) in $G_3$, the two “outer” edges are in edge orbit 3, while the “middle” edge is in edge orbit 4; 3) in $G_3$, the non-edge touching the end nodes is in non-edge orbit 2, while the two non-edges that touch the end nodes and the middle nodes are in non-edge orbit 3; 4) a node pair at node pair orbit 1 touches a $G_2$ at edge orbit 2, if it is an edge, or a $G_1$ at non-edge orbit 1, if it is a non-edge (hence, mutually exclusive edge orbit 2 and non-edge orbit 1 are reconciled into a common node pair orbit 1). There are 15 node, 12 edge, 7 non-edge, and 7 node pair orbits for up to 4-node graphlets. In a graphlet, different orbits are colored differently. All up to 5-node graphlets are used, but only up to 4-node graphlets are illustrated. There are 73 node, 68 edge, 49 non-edge, and 49 node pair orbits for up to 5-node graphlets.
Finally, node-GDV-similarity is $S(u,v) = 1 - D(u,v)$. The higher the node-GDV-similarity between nodes, the higher their topological similarity.

**Edge-GDV.** Since a graphlet contains both nodes and edges, we defined edge-GDV to count the number of graphlets that an edge touches at a given “edge orbit” (Figure 4.1) [194]. Given the automorphism group of graph $X$, $\text{Aut}(X)$, if $xy$ is an edge of $X$, the edge orbit of $xy$ is $\text{Orb}_e(xy) = \{zw \in E(X) | z = f(x) \text{ and } w = f(y) \text{ for some } f \in \text{Aut}(X)\}$, where $E(X)$ is the set of edges of $X$. There are 68 edge orbits for 3-5-node graphlets [194]. (We designed edge-GDV-similarity to measure topological similarity of edges, which we used for network clustering [194]. However, we do not use this measure for LP.)

4.2.3.2 Conceptually Novel Measures of Topology

We need to predict the existence of a link between nodes independent on whether there is an edge between them in the original network. Thus, in addition to describing the network position of an edge, we need to be able to describe the position of a non-edge as well. Hence, we generalize edge-GDV into non-edge-GDV to measure the topological position of a non-edge. Then, we reconcile mutually exclusive edge-GDVs and non-edge-GDVs into a new node-pair-GDV measure, which counts the number of graphlets that a node pair (an edge or a non-edge) touches at a given “node pair orbit” (defined below). Finally, based on node-pair-edge-GDV of a node pair, we create a new measure of the topological centrality of the node pair, called node-pair-GDV-centrality. According to this measure, the more graphlets the two nodes participate in (or share), the higher their centrality. Then, node-pair-GDV-centrality is used as a LP measure to score potential edges between node pairs in the network. The measures are defined as follows.

**Non-edge-GDV.** Analogous to edge-GDV, in this study, we define non-edge-GDV to count the number of graphlets that a non-edge touches at a given “non-edge orbit”
We define non-edge orbits as follows. If \( xy \) is a non-edge of graph \( X \), then the non-edge orbit of \( xy \) is \( \text{Orb}_{ne}(xy) = \{zw \in C(X)|z = f(x) \text{ and } w = f(y) \text{ for some } f \in \text{Aut}(X)\} \), where \( C(X) \) is the set of all non-edges of \( X \). For example, in Figure 4.1 in graphlet \( G_1 \), the only non-edge is in non-edge orbit 1. Graphlet \( G_2 \) has no non-edges. In graphlet \( G_3 \), the non-edge that touches the two end nodes is in one non-edge orbit (non-edge orbit 2), while the remaining two non-edges that touch the end nodes and the middle nodes are in a different non-edge orbit (non-edge orbit 3). And so on. There are 49 non-edge orbits for 3-5-node graphlets.

**Node-pair-edge-GDV.** Edge and non-edge orbits are mutually exclusive (Figure 4.1). However, to perform LP, we need to contrast the topological neighborhood of nodes \( v \) and \( u \) against the neighborhood of nodes \( s \) and \( t \), while hiding the information about whether \( v \) and \( u \) or \( s \) and \( t \) are actually linked. Hence, we need to reconcile edge orbits and non-edge orbits by defining node-pair-GDV to count the number of graphlets that a general node pair, which can be either an edge or a non-edge, touches at a given “node pair orbit”. For example, in Figure 4.1 a node pair at node pair orbit 1 touches a triangle (graphlet \( G_2 \)) at edge orbit 2, if the node pair is an edge, or it touches a three-node path (graphlet \( G_1 \)) at non-edge orbit 1, if the node pair is a non-edge. Hence, we reconcile mutually exclusive edge orbit 2 and non-edge orbit 1 into a common node pair orbit 1. We do this for all edge- and non-edge orbits, resulting in 49 node pair orbits for 3-5-node graphlets.

**Node-pair-GDV-centrality.** We design node-pair-GDV-centrality to assign high centrality values to node pairs that participate in many graphlets. For nodes \( v \) and \( u \), if \( c_i \) is the \( i^{th} \) element of node-pair-GDV of the two nodes, then \( \text{node-pair-GDV-centrality}(vu) = \sum_{i=0}^{49} w_i \times \log(c_i + 1) \). Thus, the more graphlets a node pair participates in, the higher its centrality. Note that we previously designed an analogous measure of the network centrality of a node, called node-GDV-centrality.
4.2.3.3 Using the New Measures for LP

Node-GDV-similarity and node-pair-GDV-centrality measures allow for several simple modifications which could perhaps improve LP accuracy, as follows.

**Combining node-GDV-similarity and node-pair-GDV-centrality.** Node-GDV-similarity favors linking topologically similar nodes. Node-pair-GDV-centrality favors linking nodes that share many graphlets. Combining the two would favor linking nodes that are both topologically similar and share many graphlets. We combine them as: \((1 - \alpha) \times \text{node-GDV-similarity} + \alpha \times \text{node-pair-GDV-centrality}\). We vary \(\alpha\) from 0 to 1 in increments of 0.2.

**Prioritizing dense graphlets.** Node-pair-GDV-centrality, as defined above, counts the number of graphlets that two nodes share, while assigning weights to different graphlets only with respect to “orbit dependencies” (see [130] for details). However, it ignores any information about the denseness of the graphlets that the two nodes share. Analogous to Adamic-Adar which favors some shared neighbors over others based on their degrees (see above), we might want to favor some shared graphlets over others based on their denseness. For example, it might be more reasonable to link two nodes that share many 4-node cliques than two nodes that share many 4-node paths. So, we favor denser shared graphlets over sparser shared graphlets by defining density-weighted (or simply weighted) node-pair-GDV-centrality (Appendix Section [C.1]). We evaluate both unweighted and weighted node-pair-GDV-centrality measures.

**Graphlet size.** To test how much of network topology is beneficial for LP, when using the graphlet-based measures, we use: 1) all 3-5-node graphlets, 2) 3-4-node graphlets, but not 5-node graphlets, and 3) only 3-node graphlets. Note that using the only 3-node graphlet within the node-pair-GDV-centrality at \(\alpha\) of 1 (see above) is equivalent to the SN measure (see above). Hence, SN is a variation of node-pair-GDV-
centrality. Also, note that when using 3-node graphlets, unweighted and weighted node-pair-GDV-centralities are equivalent. This is because there is only one 3-node graphlet when dealing with node-pair-GDVs, and its density is one. Determining which amount of topology to use is important: the more topology (the larger the graphlets), the higher the computational complexity. Exhaustive counting of all graphlets on up to \( n \) nodes in graph \( G(V, E) \) takes \( O(|V|^n) \); but, the practical running time is much smaller due to the sparseness of real networks [124, 131, 166]. Also, counting is embarrassingly parallel. Finally, fast non-exhaustive approaches exist for counting graphlets [124].

4.2.4 Evaluation Framework

We evaluate each of the existing and new LP methods on each of the PPI networks as follows.

First, we introduce synthetic noise in the given PPI network by randomly removing 5%-50% of its edges, with the goal of measuring how well the different methods can reconstruct the original network, using the original PPIs as the ground truth data. We apply the given LP measure to a “noisy” network created in this way and score each node pair in the network, so that the higher the score, the more likely the nodes are to be linked. We predict \( k\% \) of the highest-scoring node pairs as edges. We vary \( k \) from 0% to 100% in increments of 1%. At each \( k \), we count the number of true positives, true negatives, false positives, and false negatives, and we compute: 1) precision, recall, and F-score; and 2) sensitivity and specificity (Appendix Section C.2 [43]. For simplicity of comparing results across different methods, we summarize the performance of the methods over the entire range of \( k \) with respect to sensitivity and specificity by calculating the areas under receiver-operator curves (AUROCs), as well as with respect to precision and recall by calculating the areas under precision-recall curves (AUPRs).
To account for randomness in the above procedure, for each level of noise, we randomly remove the given percentage of edges from the original network five times and average the above statistics over the five runs. Ideally, we would perform more random runs, but this is impractical due to the required computational time. Plus, this might be unnecessary, since the standard deviations resulting from the five runs are typically very small (Section 4.3), and since even with five random runs of each method, we can compute the statistical significance of the difference in LP accuracy between a pair of methods by using the paired $t$-test. With this test, we compare five pairs of AUROCs corresponding to five random runs of two methods, and a low $p$-value would indicate that the null hypothesis (the difference between the accuracy of the two methods having a mean of 0) can be rejected.

Second, due to the availability of low-confidence PPI data for the HC network (see above), we perform an additional evaluation test: we apply the given LP method to the HC network and use the low-confidence PPIs as the ground truth data. We evaluate the method in the same way as above.

Third, we apply the given LP method to a network to de-noise it, and we evaluate the biological quality of the de-noised network with respect to the “enrichment” of predicted edges in Gene Ontology (GO) terms [9]. We compute the enrichment as the percentage of predicted edges, out of all edges in which both proteins have at least one GO term, in which the two end nodes share a GO term. As [10], we do this for biological process GO terms. To avoid potential biases, we consider only gene-GO term associations with experimental evidence codes. Since we de-noise networks by relying on their topology (i.e., on the PPIs), to avoid “circular arguments”, of these associations, we exclude associations inferred from PPIs. We compute the statistical significance of the enrichment by using the hypergeometric model (Appendix Section C.2).

Finally, we validate predicted edges absent from the original network by searching
for them in an independent PPI data source. Here, we use BioGRID [24], because it is a trusted PPI data source. Again, we measure the statistical significance of validating the given number of predictions by using the hypergeometric model (Appendix Section C.2). We perform the external data source validation on AP/MS predictions as this network uses the same naming scheme as BioGRID.

4.3 Results and Discussion

We study three yeast PPI networks: AP/MS, Y2H, and HC. We use a number of existing and new LP measures. The existing measures are degree product (DP), shared neighbors (SN), Jaccard coefficient (JC), Adamic-Adar (AA), Katz index (Katz), local path index (LPI), resource allocation index (RAI), and random walk with resistance (RWS). The new measures are node-GDV-similarity and node-pair-GDV-centrality. See Section 4.2 for details.

The two new graphlet-based measures allow us to address several important LP questions. First, we can combine node-GDV-similarity, which favors linking nodes with topologically similar neighborhoods, with node-pair-GDV-centrality, which favors linking nodes that share many graphlets and thus have large extended shared neighborhoods, to favor linking nodes that are both topologically similar and share many graphlets, which might be preferred. To test whether this is the case, we combine the two measures by varying the value of parameter $\alpha$ from 0 to 1, where $\alpha$ of 0 means that only node-GDV-similarity is used, and $\alpha$ of 1 means that only node-pair-GDV-centrality is used (see Section 4.2). Second, we test whether favoring denser graphlets that are shared between the nodes in question within the node-pair-GDV-centrality measure is preferred over equally favoring all graphlets, independent on their density. We do this by evaluating both unweighted and weighted node-pair-GDV-centrality measures (see Section 4.2). Third, to test how much of network topology is beneficial for LP, when using the graphlet-based measures, we use: 1) all
3-5 node graphlets, 2) only 3-4-node graphlets, and 3) only 3-node graphlets.

After we compare the different variations of graphlet-based measures, we evaluate the best of them against the existing LP measures. We perform three evaluation tests: 1) we introduce synthetic noise in the given PPI network by randomly removing a percentage of its edges, with the goal of measuring how well the given method can reconstruct the original network, using the original PPIs as the ground truth data; 2) given the availability of low-confidence PPI data corresponding to the HC network, we apply the given LP method to the original HC PPI network and use the corresponding low-confidence PPI data as the ground truth data when evaluating the method; and 3) we apply the LP methods to the original PPI networks to de-noise them, and we evaluate biological quality of the de-noised networks. In the first two evaluation tests above, we use systematic AUROC and AUPR frameworks as the evaluation criteria (Section 4.2). In the third evaluation test above, we compute the “enrichment” of predicted edges in GO terms [9], and also, we validate predicted edges in an external data source (Section 4.2).

Ultimately, we are less focused on identifying a superior LP method but more on testing whether we can de-noise a network so that the de-noised network is biologically more meaningful than the original one, as well as on which topological properties affect LP accuracy.

4.3.1 Test 1: Evaluating LP Methods by Introducing Synthetic Noise into PPI Networks

Current PPI networks are noisy. The correct and complete ground truth interactomes are unknown. Thus, an alternative ground truth data has to be sought. We create synthetic ground truth data from the real PPI networks. For each PPI network, we add synthetic noise to the network by randomly removing 5%, 10%, 15%, 20%, 25%, and 50% of the original edges. Then, we evaluate the given LP method
by applying it to a synthetically noised network and by measuring how well it reconstructs the original network (Section 4.2). Below, we initially discuss our results in the context of AUROCs, and later on we contrast these results against those returned by AUPRs.

4.3.1.1 Combining Topological Similarity and Centrality of Nodes to be Linked Improves LP Accuracy

By combining node-GDV-similarity and node-pair-GDV-centrality with parameter $\alpha$ (Section 4.2), we find that nodes that are simultaneously topologically similar and share many graphlets are preferred for LP. In general, the larger the value of $\alpha$ (the more node-pair-GDV-centrality is used), the better the LP accuracy (Figure 4.2(a)). This suggests that the topological similarity of two nodes is less relevant for LP than the number of graphlets that the nodes share. However, using a small amount of node-GDV-similarity in the combined LP score ($\alpha = 0.8$) actually improves LP accuracy compared to using node-pair-GDV-centrality alone ($\alpha = 1$), implying that topological similarity is relevant. The difference between LP accuracy of the best $\alpha$ of 0.8 and any other $\alpha$ in Figure 4.2(a) is statistically significant, with $p$-values below $2.5 \times 10^{-6}$ for 5% noise and below $2.8 \times 10^{-4}$ for 50% noise.

While the results in Figure 4.2(a) are for weighted node-pair-GDV-centrality, 3-5-node graphlets, two noise levels, and the AP/MS network, in general, they also hold for weighted node-pair-GDV-centrality, all graphlet sizes, all noise levels, and HC and Y2H networks (Appendix Figures C.1 and C.2). And since $\alpha = 0.8$ is statistically significantly superior to all other $\alpha$s, in the rest of the section, we focus only on this value of $\alpha$. 

92
Figure 4.2: LP accuracy of the graphlet-based methods in the context of evaluation test 1. The accuracy is shown in terms of AUROCs (panels (a), (c), and (e)) as well as AUPRs (panels (b), (d), and (f)) at the lowest noise level of 5% and the highest noise level of 50% when comparing: 1) varying \( \alpha \) s in the AP/MS network (panels (a) and (b)); 2) unweighted (“u”) vs. weighted (“w”) node-pair-GDV-centrality in the HC network (panels (c) and (d)); and 3) different graphlet sizes (3-5-node (“g3-5”), 3-4-node (“g3-4”), and 3-node (“g3”) graphlets) in the Y2H network (panels (e) and (f)). Note that we intentionally vary the networks between panels (AP/MS in panels (a) and (b), HC in panels (c) and (d), and Y2H in panels (e) and (f)), but only in order to represent each of the three studied networks equally; we show full results throughout the Appendix C.
4.3.1.2 Favoring Denser Shared Graphlets Improves LP Accuracy

We find that preferring denser graphlets (Section 4.2) improves the LP performance: weighted node-pair-GDV-centrality outperforms the unweighted version (Figure 4.2(c)), and its superiority is statistically significant, with \( p \)-value of \( 1.76 \times 10^{-8} \) for 5% noise and \( 1.8 \times 10^{-5} \) for 50% noise.

While the results in the figure are for all 3-5-node graphlets, two noise levels, and the HC network, in general, they also hold for all graphlet sizes, noise levels, and AP/MS and Y2H networks (Appendix Figure C.3). Thus, henceforth, we focus only on the superior weighted version of node-pair-GDV-centrality.

4.3.1.3 Using More Topology Does Not Always Guarantee higher LP Accuracy

There is no clear trend on how much topology is best (Figure 4.2(e)). For example, in the figure, for the lowest noise level of 5%, using 3-5-node graphlets is statistically significantly superior over using 3-4-node or 3-node graphlets, with \( p \)-values of \( 5.8 \times 10^{-3} \) and \( 1.1 \times 10^{-2} \), respectively. On the other hand, for the highest noise level of 50%, using 3-5-node graphlets is marginally superior over using 3-4-node graphlets, with \( p \)-value of \( 5.6 \times 10^{-2} \), and it is statistically significantly superior over using 3-5-node graphlets, with \( p \)-value of \( 8.6 \times 10^{-3} \). Hence, using more network topology can improve LP accuracy, but it is not guaranteed to do so.

Whereas the results in Figure 4.2(e) are only for two noise levels and the Y2H network, in general, they also hold for other noise levels and for AP/MS and HC networks (Appendix Figure C.4).

Because in some cases using only 3-node graphlets is superior, and because the existing shared neighbors-based methods and SN in particular also rely on 3-node graphlets (Section 4.2), one might incorrectly assume that in these cases, our graphlet-based methods do not improve upon the existing SN method. However, it is at \( \alpha \) of 1 when our node-pair-GDV-centrality and existing SN method are equivalent.
Since our results at $\alpha$ of 0.8 are superior over results at $\alpha$ of 1 (see above), and since at $\alpha$ of 0.8 SN is actually combined with graphlet information encoded in the node-GDV-similarity measure, node-GDV-similarity actually *improves* the accuracy of SN even when using 3-node graphlets only and especially when using all 3-5-node graphlets is superior to using only 3-node graphlets.

Also, using deeper network topology is superior to using only the direct network neighborhood of nodes to be linked in the sense that node-GDV-similarity alone ($\alpha = 0$) is superior to the existing DP method (Appendix Figure C.5). This is interesting because the two methods are somewhat similar. They both take into account graphlet degrees of two nodes in question. They differ in that DP considers only the 2-node graphlet and hence captures only the direct (1-deep) network neighborhoods of the nodes, whereas node-GDV-similarity considers all 2-5-node graphlets, thus capturing up to 4-deep node neighborhoods. Hence, in this context, including more network topology helps. (We compare the existing methods to our new graphlet-based methods in more detail in the following section.)

In general, using larger graphlets can increase LP accuracy (the following sections also confirm this). Since counting larger graphlets is computationally expensive compared to counting smaller graphlets (Section 4.2), whether it is worth including the extra topological information that is captured by the larger graphlets depends on how significant the improvement is.

4.3.1.4 New Graphlet-based Measures are Superior to the Majority of the Existing Measures

Having examined different variations of the graphlet-based measures, we now compare these measures with the existing ones. Of all graphlet-based variations, we report the weighted version at $\alpha = 0.8$ when considering 3-5-node graphlets, since this version generally performs the best. The existing measures include DP, SN, JC,
Figure 4.3: Comparison of different methods in the context of evaluation test 1. Our best method (“ours”) is compared against existing methods (DP, SN, JC, AA, Katz, LPI, RAI, and RWS) in terms of AUROCs (panel (a)) and AUPRs (panel (b)) for synthetically noised AP/MS, HC, and Y2H networks at 5% noise level. Here, “ours” corresponds to using 3-5-node weighted graphlets at $\alpha = 0.8$. Results for all other noise levels are shown throughout the Appendix C.

AA, Katz, LPI, RAI, and RWS.

Our graphlet-based method is superior to five out of the eight existing methods in all three networks (Figure 4.3(a)). These five methods are: DP, SN, JC, AA, and RAI. And whereas Katz, LPI, and RWS are somewhat superior to our graphlet-based method in this evaluation test and with respect to the evaluation criterion from Figure 4.3(a) (namely AUROCs), we show later that our method beats each of Katz, LPI, and RWS in at least one different evaluation test or with respect to at least one other evaluation criteria (such as AUPRs or biological correctness of de-noised networks; see below). The $p$-value of the difference between LP accuracy of our method and any of the five inferior methods in Figure 4.3(a) is below $8.6 \times 10^{-6}$, $1.5 \times 10^{-6}$, and $1.2 \times 10^{-6}$ for AP/MS, HC, and Y2H networks, respectively. It is worth noting that most of the methods perform quite well, reaching very high AUROCs of up to 0.0988, 0.998, and 0.98 in AP/MS, HC, and Y2H networks, respectively. Whereas Figure 4.3(a) is for the lowest noise level, the results are similar for other noise levels (Appendix Figure C.5). Interestingly, our graphlet-based measures further improve
over the existing measures as the noise increases.

4.3.1.5 ROCs vs. Precision-Recall Curves

Thus far, we have shown AUROC results. AUROCs are commonly used to evaluate methods over the entire [0%, 100%] range of $k$ (Section 4.2 and Appendix Figures C.1-C.8). In a similar fashion, AUPRs can be computed (Section 4.2 and Appendix Figures C.9-C.19). Whereas one would hope that different evaluation criteria such as AUROCs and AUPRs would identify the same methods as best performing, we actually find that the AUROC results are not necessarily consistent with AUPR results.

Namely, while in both AP/MS and HC networks our best graphlet-based measure is better than JC, AA, and RAI with respect to AUROCs (along with some other existing measures) (Figure 4.3(a) and Appendix Figure C.5), JC, AA, and RAI are better with respect to AUPRs (Figure 4.3(b) and Appendix Figure C.13). On the other hand, whereas Katz, LPI, and RWS are better than our graphlet-based method with respect to AUROCs (Figure 4.3(a)), our graphlet-based method is better than LPI and RWS with respect to AUPRs (Figure 4.3(b)). Note that while Katz remains superior to our graphlet-based methods with respect to AUPRs, as we show later, Katz becomes inferior to our graphlet-based methods with respect to an alternative evaluation criterion, namely biological correctness of de-noised networks (see below).

Further, for Y2H, we notice a different inconsistency: whereas AUROCs are high for the best-performing methods (Appendix Figure C.5), AUPRs indicate poor performance of almost all methods, as precision is always relatively low (Appendix Figure C.13).

Even though optimizing AUROCs does not necessarily optimize AUPRs, the observed inconsistencies are alarming, and the LP community needs to be aware. We address this issue by also comparing the different methods with respect to biological
correctness of their de-noised networks (see below). But first, we check whether results depend on the ground truth data, as follows.

4.3.2 Test 2: Evaluating LP Methods on HC Network with respect to Low-confidence PPI Data

When we evaluate the LP methods on the HC network with low-confidence PPIs as the ground truth data, we find that:

1. As in the previous section (when evaluating LP methods by introducing synthetic noise into PPI networks), combining topological similarity and centrality of nodes to be linked improves LP accuracy. However, now $\alpha$ of 0.4 is the best overall instead of $\alpha$ of 0.8 (Appendix Figures C.20 and C.21): topological similarity is now more relevant than the number of shared graphlets.

2. As in the previous section, favoring denser graphlets improves LP accuracy for the best $\alpha$ (Appendix Figure C.22).

3. As in the previous section, using more topology can improve LP accuracy (Appendix Figure C.24). Using 3-4-node graphlets at the best $\alpha$ of 0.4 results in higher AUROC than using only 3-node graphlets at any $\alpha$, and using 3-5-node graphlets at $\alpha$ of 0.8 results in higher AUROCs than using only 3-node graphlets or 3-4-node graphlets at the same $\alpha$ (Appendix Figure C.23).
4. As in the previous section, our best graphlet-based measure in this context (using 3-4-node weighted graphlets at $\alpha = 0.4$) is superior to the majority (namely six) of the eight existing measures (Figure 4.4(a) and Appendix Figure C.24). The only superior methods in this context (namely with respect to AUROCs) are Katz and LPI, but as we show later, these two methods are inferior to our graphlet-based method with respect to at least one alternative evaluation criterion.

5. We again see inconsistencies between AUROC and AUPR results (Figure 4.4 and Appendix Figures C.25-C.29). For example, whereas LPI is superior to our graphlet-based method with respect to AUROCs, LPI is inferior to our graphlet-based method with respect to AUPRs.

4.3.3 Test 3: Denoising the PPI Networks

Since both our new and the existing methods perform well on all networks with respect to AUROCs (Figures 4.3(a) and 4.4(a), and Appendix Figure C.5), we use the overall best graphlet-based method (weighted 3-5-node graphlets at $\alpha = 0.8$) as well as DP, SN, JC, AA, Katz, LPI, RAI, and RWS and to de-noise the networks. We score each node pair in a network, and we predict as edges in the de-noised network the top $k$% highest scoring node pairs. We choose $k$ so that the number of edges in the de-noised network matches the number of edges in the original network. Depending on the network, $k$ falls between 1% and 2%. We choose $k$ in this way because most of the methods achieve the maximum F-score in this range of $k$ (Appendix Figures C.17-C.19).

4.3.3.1 Gene Ontology (GO) Validation of De-noised Networks

We validate biological correctness of the de-noised networks by computing the enrichment of all predicted edges in GO terms (see Section 4.2 and Figure 4.5). When we de-noise AP/MS and HC networks, the enrichment is statistically significant for all methods ($p$-values $\leq 10^{-100}$). Our method as well as JC, AA, LPI, and RAI improve the quality of the original AP/MS and HC networks. This is important, since the main goal of LP is to de-noise a network so that the de-noised network is
Figure 4.5: Comparison of different methods in the context of evaluation test 3. Our best method (“ours”) is compared against existing methods (DP, SN, JC, AA, Katz, LPI, RAI, and RWS) in terms of GO enrichments of AP/MS, HC, and Y2H networks and their de-noised counterparts. Here, “ours” corresponds to using 3-5-node weighted graphlets at $\alpha = 0.8$.

more meaningful than the original one. While the GO enrichments are worse for the de-noised networks than for the original Y2H network for all methods but RWS, the enrichments are still statistically significant ($p$-values $\leq 0.05$) for our method, as well as for all existing methods except SN.

Importantly, in this evaluation test and with respect to this evaluation criteria, Katz, which is the only method that is superior to our best graphlet-based method with respect to both AUROCs and AUPRs, now loses its superiority: our best graphlet-based method now beats Katz in AP/MS and HC networks. This not only verifies that our method is superior to every one of the eight existing methods with respect to at least one evaluation criterion (be it AUROCs, AUPRs, or biological correctness of de-noised networks), but it also implies an additional inconsistency between the different evaluation criteria regarding the best-performing method(s). As a further illustration of this inconsistency, we note that JC and AA, which are not su-
prior to all other measures with respect to either AUROCs or AUPRs, now slightly outperform all of the other measures for AP/MS and HC networks (Figure 4.5).

4.3.3.2 Intersection of De-noised Networks Produced by Different Methods

Since we de-noise a network with multiple LP methods, we measure the intersections between the de-noised networks (Appendix Figure C.30). The intersections are quite large between our method on one side and the shared neighbors-based methods or LPI on the other. JC is an exception, as it is somewhat different not only from our method but also from other shared neighbors-based methods. Actually, our method is more similar to SN and AA than JC is. The similarity between LPI and our method, as well as between LPI and the shared-neighbors-based methods (excluding JC) is not surprising, as all of these measures are intuitively similar (Section 4.2). In terms of the intersections between the original networks on one hand and de-noised networks for the different methods on the other, the intersections are the largest for Katz, followed by a number of more-less tied methods (including ours), followed by DP (Appendix Figure C.30).

It is important to note that de-noised AP/MS and HC networks resulting from Katz (which achieves extremely high AUROCs and AUPRs in AP/MS and HC networks in evaluation tests 1 and 2) are completely identical to the original AP/MS and HC networks. That is, when used to de-noise these networks, Katz cannot generate any new edges or remove any of the existing ones (and hence the maximum overlap with the original networks in Appendix Figure C.30). And while on one hand one might argue that because of high AUROCs and AUPRs Katz is very accurate, on the other hand, Katz is clearly incapable of de-noising real-world networks, which could be viewed as its disadvantage.
4.3.3.3 Validation of De-noised Networks on External PPI Data

We aim to validate “new predicted edges” (predicted edges not present in the original network; Appendix Table C.1) by searching for them in BioGRID as an independent data source. We do this for the AP/MS network. Even though validation accuracy varies across the methods, all methods achieve statistically significant validation rates ($p$-values below $1 \times 10^{-100}$), except Katz, which predicts no new edges and thus cannot be validated. Of the remaining existing methods, only RAI, JC, and RWS outperform our method (Appendix Figure C.31).

4.4 Conclusions

We tackle the problem of link prediction (LP) in the context of PPI network de-noising. We comprehensively study what is it in the PPI network topology around nodes in question that dictates whether the nodes should be linked. To evaluate whether nodes that share many neighbors and are thus close in the network are favored over distant nodes (as is the assumption of most of the existing LP methods), whether topological similarity between nodes in question has any effect, and how much of the network topology should be included, we propose new LP methods, since none of the existing methods allowed for answering all of these questions. Unlike the existing methods, our new methods allow for combining topological similarity of the nodes to be linked with the information about the size of their shared neighborhood, while at the same time allowing to vary the amount of network topology that is taken into account for LP. We demonstrate via a thorough evaluation that our new methods outperform each of the eight existing methods with respect to at least one evaluation criterion. Importantly, when we use the LP methods to de-noise real-world PPI networks, we find that the de-noised networks improve biological correctness of the original networks, which is the ultimate goal of LP in computational biology.
4.5 Computational Novelty of the Work

We propose a novel LP methodology that not only accounts for extended topology but also allows for a fair evaluation of important LP parameters. Unlike the existing methods, our new methods allow for combining topological similarity of the nodes to be linked with the information about the size of their shared neighborhood, while at the same time allowing to vary the amount of network topology that is taken into account for LP. In particular, we validate our hypothesis that nodes that are topologically similar and have large and dense extended shared neighborhoods should be linked together. However, we also show that including more network topology does not always improve LP accuracy. We demonstrate via a thorough evaluation that our new methods outperform each of the eight existing methods with respect to at least one evaluation criterion. Alarming, we find that different LP evaluation criteria disagree in identifying the most accurate method(s), which has important implications in any LP domain. Importantly, when we use the LP methods to de-noise real-world PPI networks, we find that the de-noised networks improve biological correctness of the original networks, which is the ultimate goal of LP in computational biology. For example, when our link prediction was used to computationally study human aging from PPI networks (which needs to be done because human aging is hard to study experimentally due to long lifespan and ethical constraints), this improved the quality of aging-related predictions compared to using the original noisy data. We have made publicly available our implementations of the methods, network data we used (including randomized data), and the list of all node pairs predicted as edges by any of the considered methods.\footnote{http://www.nd.edu/~cone/GLP}
4.6 My Contributions to the Work

I helped implement the methods, helped carry out the computational analyses, helped analyze the results, and helped write the manuscript.
CHAPTER 5

EXPLORING THE STRUCTURE AND FUNCTION OF TEMPORAL NETWORKS WITH DYNAMIC GRAPHLETS

The work presented in this chapter has been published as follows:


5.1 Introduction

5.1.1 Motivation

Networks (or graphs) are powerful models of complex systems in various domains, from biological cells to societies to the Internet. Traditionally, due to limitations of data collection techniques, researchers have mostly focused on studying the static network representation of a given system [145]. However, many real-world systems are not static but change over time [78]. With new technological advancements, it has become possible to record temporal changes in network structure (or topology), corresponding to arrival or departure times of nodes or edges. Examples of temporal networks include cellular [169], functional brain [206], person-to-person communication [163], online social [112], or citation [111] networks.

The increasing availability of temporal real-world networks, while opening new opportunities, has also raised new challenges for researchers. Namely, despite a large arsenal of powerful methods that already exist for studying static networks, these
methods cannot be directly applied to temporal networks. Instead, the simplest approach to deal with a temporal network is to completely discard its time dimension by aggregating all nodes and edges from the temporal data into a single static network. While this would allow to directly apply to the resulting aggregate network the existing and well-established methods for static network analysis, such an aggregate or static approach loses all important temporal information from the data. To overcome this, one could model the temporal network as a series of snapshots, each of which is a static network that aggregates the temporal data observed during the corresponding time interval. Then, with such a snapshot-based network representation, one could use a static-temporal approach to study each snapshot independently via the existing methods for static network analysis and then consider time-series of the results. However, this strategy treats each network snapshot in isolation and discards relationships between the different snapshots. Clearly, both static and static-temporal approaches overlook temporal information that is important for studying a dynamic system [78]. Therefore, proper analysis of temporal networks requires development of novel strategies that can fully exploit the temporal information from the data. This is the focus of our study.

5.1.2 Related Work

**Static networks.** One way to study the structure of a static network is to compute its **global** properties such as the degree distribution, diameter, or clustering coefficient [103, 145]. However, although global network properties can summarize the structure of the entire network in a computationally efficient manner, they are not sensitive enough to capture detailed topological characteristics of the complex real-world networks [163]. Thus, **local** properties have been proposed that can capture more detailed aspects of complex network structure. For example, one can study small **partial** subgraphs called **network motifs** that are statistically significantly over-
represented in a network compared to some null model [136, 137]. The practical usefulness of network motifs has been questioned, since the choice of null model can significantly affect the results [8], and since selecting an appropriate null model is not a trivial task [132]. To address this issue, graphlets have been proposed [167], which are small induced subgraphs of a network that can be employed without reference to a null model (Figure 5.1(a)), unlike network motifs. Also, unlike network motifs, graphlets must be induced subgraphs, whereas motifs are partial subgraphs, which makes graphlets more precise measures of network topology compared to motifs [164].

Graphlets have been proven in static network research. They were used as a basis for sensitive measures of network [166, 167] or node [130] similarities. These measures in turn have been used to develop state-of-the-art algorithms for many computational problems such as network comparison [121, 122, 205, 218, 222], alignment [80, 102, 134, 181], clustering [83, 194], or de-noising [84], as well as for various application problems in computational biology, such as studying human aging [54, 55], cancer [75, 133] and other diseases [214], pathogenicity [135, 194], or receptor-ligand interactions [191].

**Temporal networks.** Just as static networks, temporal networks can be studied by considering evolution of their global properties [111, 151]. Since this again leads to imprecise insights into network changes with time, recent focus has shifted onto local-level dynamic network analysis via notion of “temporal motifs”. In the simplest case of the static-temporal approach, static motifs are counted in each snapshot and then their counts are compared across the snapshots [23]. To overcome this approach’s limitation of ignoring any motif relationships between different snapshots, the notion of static network motifs has been extended into several notions of temporal motifs [12, 30, 88, 97, 98, 231]. However, the temporal motif approaches suffer from the following drawbacks. 1) They only deal with motif structures of limited size or topological complexity (e.g., linear paths) [12, 88, 231], which limits their practical
usefulness to capture complex network structure in detail. 2) They pose additional constraints, such as limiting the number of events (temporal edges) a node can participate in at a given time point [97, 98]. 3) They allow for obtaining the motif-based topological “signature” of the entire network only but not of each individual node [12, 23, 30, 88, 97, 98, 231], whereas the latter is useful when aiming to link the network topological position of a node to its function via e.g., network alignment or clustering. 4) Importantly, like static motifs, all temporal motif approaches rely on a null model, which again questions their practical usefulness [8, 131, 132], especially since choosing an adequate null model is even harder in the dynamic than static setting.

Analogous to extending the notion of network motifs from the static to dynamic setting, recently, we used graphlets as a basis of a static-temporal approach to study human aging from biological networks [54]. We counted static graphlets within each snapshot (corresponding to a given human age), and then we studied the time-series of the results to gain insights into network structural changes with age [54]. In this initial work, we only used the static graphlets within a static-temporal approach that ignored important relationships between different snapshots, in order to demonstrate that accounting for at least some temporal information in the static-temporal fashion can improve results compared to using the traditional static (aggregate) approach. Further important temporal inter-snapshot information remains to be explored via a novel truly temporal approach. We aim to develop such an approach, as follows.

5.1.3 Our Approach and Contributions

To overcome the issues of the existing methods for temporal network analysis, we take the well-established static graphlets to the next level to develop new theory of dynamic graphlets that allow for efficient temporal analysis. Unlike the existing temporal motif approaches, our dynamic graphlets allow for all of the following.
1) They can study topological and temporal structures of arbitrary complexity, as permitted by available computational resources. 2) There are no limitations on e.g., the number of events that a node can participate in. 3) They can capture the topological signature of the entire network and of each individual node. 4) They allow for studying temporal networks without relying on a null model. Also, unlike the existing static-temporal graphlet approach, dynamic graphlets explicitly account for inter-snapshot relationships.

Of the existing methods, the closest to our work are temporal motifs as defined in [97], static graphlets [130, 167], and static-temporal graphlets [54]. Since temporal motifs depend on a null model and have other limitations, they cannot be directly and fairly compared to our dynamic graphlets. Static and static-temporal graphlets are directly comparable to our dynamic graphlets. By comparing the different graphlet approaches, we can fairly evaluate the effect on result accuracy of the amount of temporal information that each graphlet approach can consider.

In the rest of the chapter, we formally define our novel notion of dynamic graphlets (Section 5.2). We thoroughly evaluate their ability to characterize the structure and function of an entire temporal network as well as of individual nodes. Namely, on both synthetic and real-world temporal network data, we measure how well our approach can group (i.e., cluster) temporal networks (or nodes) of similar structure and function and separate dissimilar networks (or nodes). We find that our dynamic graphlet approach outperforms both static and static-temporal graphlet approaches in all of these tasks (Section 5.3). This confirms our hypothesis that accounting for more temporal information helps. We demonstrate one possible application of dynamic graphlets: to study age-specific structural and functional changes in the cell from temporal aging-related molecular network data of human (Section 5.3.4).
5.2 Methods

We introduce dynamic graphlets in Section 5.2.1, give an algorithm for their counting in Section 5.2.2 and describe our evaluation in Section 5.2.3.

5.2.1 Dynamic Graphlets

Let $G(V, E)$ be a temporal network, where $V$ is the set of nodes and $E$ is the set of events (temporal edges) that are associated with a start time and duration. An event can be represented as a 4-tuple $(u, v, t_{start}, \sigma)$, where $u$ and $v$ are its endpoint nodes, $t_{start}$ is its starting time, and $\sigma$ is its duration. Thus, each event is linked to a unique edge in the aggregate static network, whereas each static edge may be linked to multiple events with different starting times. Note that here we consider undirected events, but most ideas can be extended to directed events as well.

Given a temporal network (as defined above), our goal is to extend the notion of static graphlets in order to capture how the network neighborhood of a node changes over time. For example, since multiple events can occur over the same edge, the same static graphlet $G_1$ (Figure 5.1(a)) can correspond to multiple dynamic graphlets (defined below), such as $D_4$ and $D_6$ (Figure 5.1(b)), depending on the order of events in the temporal network. For this reason, we aim to develop methodology that allows for distinguishing between such different temporal network structures.

To formalize this desired intuition of dynamic graphlets, we first introduce the notion of a time-respecting path, whose goal is to connect two nodes so that for any two consecutive events in the path, the later event starts no later than $\Delta t$ time after the earlier event ends (i.e., so that the two events are $\Delta t$-adjacent). Formally, we say that two nodes $s$ and $d$ are connected by a $\Delta t$-time-respecting path if there is a sequence of events $(v_0, u_0, t_0, \sigma_0), (v_1, u_1, t_1, \sigma_1), \ldots, (v_k, u_k, t_k, \sigma_k)$, such that $v_0 = s$, $u_k = d$, $\forall i \in [0, k - 1]$ $u_i = v_{i+1}$, and $t_{i+1} \in [t_i + \sigma_i, t_i + \sigma_i + \Delta t]$. The above $\Delta t$ constraint allows a user to control how much time (at most) can pass between two
events so that they are considered to be consecutive (i.e., $\Delta t$-adjacent).

Given the above definitions, a temporal network is called $\Delta t$-connected if for any (unordered) pair of nodes there exists a $\Delta t$-time-respecting path between the two nodes. Also, we define a $G'(V', E')$ to be a temporal subgraph of $G$ with $V' \subseteq V$ and $E' \subseteq E$, where $E'$ is restricted to nodes in $V'$. Then, a dynamic graphlet is an equivalence class of isomorphic $\Delta t$-connected temporal subgraphs; equivalence is taken with respect to the relative temporal order of events, without considering the events’ actual start times. Hence, two $\Delta t$-connected temporal subgraphs will correspond to the same dynamic graphlet if they are topologically equivalent and their corresponding events occur in the same order. Figure 5.1(b) illustrates all dynamic graphlets with up to three events, but we evaluate larger graphlets as well.

Note that if for a given dynamic graphlet with $n$ nodes and $k$ events we discard the order of the events and remove duplicate events over the same edge, we get a static graphlet with $n$ nodes and $k' \leq k$ edges (e.g., $G_1$ from $D_6$), which we call the backbone of the dynamic graphlet. Each dynamic graphlet has a single backbone, while one backbone can correspond to different dynamic graphlets (Figure 5.1(c) and Appendix Table D.1).

The above definitions allow us to describe all dynamic graphlets of a given size in the entire network, in order to obtain topological signature of the network. There already exists a popular notion of topological signature of an individual node in a static network, called the graphlet degree vector (GDV) of the node, which describes the number of each of the static graphlets that the node “touches” at a specific “node symmetry group” (or automorphism orbit) (Figure 5.1(a)) \cite{I30}. Here, we aim to describe the node’s dynamic GDV equivalent. In this case, automorphism orbits of a dynamic graphlet will be determined based on both topological (as in static case) and temporal (unlike in static case) position of a node within the dynamic graphlet. Thus, a dynamic graphlet with $n > 2$ nodes will have $n$ different orbits.
Figure 5.1: Illustration of the difference between static and dynamic graphlets. (a) All nine static graphlets with up to four nodes, along with their 15 “node symmetry groups” (or formally, automorphism orbits) \[130, 167\]. Within a given graphlet, different orbits are denoted by different node colors. For example, there is a single orbit in graphlet \(G_2\), as all three nodes are topologically identical to each other. But there are two orbits in graphlet \(G_1\), as the two end nodes are topologically identical to each other but not to the middle node (and vice versa). (b) All dynamic graphlets with up to three events, along with their automorphism orbits. Multiple events along the same edge are separated with commas. Node colors correspond to different orbits. (c) All four dynamic graphlets \(D_i\) whose static backbone is \(G_1\).
(Figure 5.1(b)), whereas the number of orbits in a static graphlet of size \( n \) is typically less than \( n \) (Figure 5.1(a)); only for a dynamic graphlet with \( n = 2 \) (e.g., \( D_3 \)), there will be only one orbit, since events are undirected and thus the two end nodes in such a graphlet are topologically equivalent.

Next, we aim to compute \( D(n, k) \), the number of dynamic graphlet types with \( n \) nodes and \( k \) events. Since at least \( n - 1 \) edges are needed to connect \( n \) nodes, it follows that \( D(n, k) = 0 \) for \( k < n - 1 \). Further, since our events are undirected, if follows that \( D(2, k) = 1 \), for any \( k \). To compute \( D(n, k) \) when \( n \geq 3 \) and \( k \geq n - 1 \), notice that each dynamic graphlet with \( k \) events can be formed from a dynamic graphlet with \( k - 1 \) events and either \( n - 1 \) or \( n \) nodes, by adding a new event between some two existing nodes or between an existing node and a new node, respectively (Figure 5.2). From these observations, we get the following recursive formulas for \( D(n, k) \): \( D(3, k) = 3D(3, k - 1) + D(2, k - 1) \), \( n = 3 \), and \( D(n, k) = (2n - 3)D(n, k - 1) + 2D(n - 1, k - 1) \), \( n > 3 \), plus the corresponding closed-form solution (Appendix Section D.1 and Appendix Table D.2):

\[
D(n, k) = \sum_{i=0}^{n-2} \frac{(-1)^{n+i}(n-2)^i(2i+1)^{k-1}}{2(n-2)!}, \ n \geq 3.
\]

Since now we can compute \( D(n, k) \), we next consider the task of enumerating and generating each of these dynamic graphlet types (in Section 5.2.2, we discuss the process of counting each of the generated graphlets in a given network). We build upon the fact that each dynamic graphlet with \( k \) events has a unique \((k - 1)\)“prefix” (see above). Thus, we start with a single event (dynamic graphlet \( D_0 \) with \( n = 2 \) and \( k = 1 \)) as the current graphlet and then recursively extend the current graphlet until the desired size is reached. Appendix Algorithm D.1 illustrates our enumeration procedure.
Figure 5.2: Illustration of how we extend a dynamic graphlet with an additional more recent event, on the example of $D_9$. There are seven possible extensions of $D_9$ (which contains four nodes and three events) with the most recent event 4 (shown in bold) into a dynamic graphlet with four events. Five of the extensions keep the same number of nodes but increment the number of events, while the remaining two extensions increment both the number of nodes and events. Note that in order to extend $D_9$ with event 4, at least one of the nodes involved in event 3 has to participate in event 4 as well.

5.2.2 Counting Dynamic Graphlets in a Network

As now we know the number of dynamic graphlet types of a given size and how to enumerate and generate each one of them, how to actually count each of the dynamic graphlets in a given network? Here, we discuss key ideas behind our counting procedure; for a detailed description and the corresponding algorithms, see Appendix Section D.2.

We perform dynamic graphlet counting in the same way as we generate the graphlet types. That is, for each event in a temporal network, we use this event as the current dynamic graphlet (of type $D_0$) and then search for larger graphlets that are grown recursively from the current one (Figure 5.3). Appendix Algorithms D.2–D.4 describe this procedure. Its running time depends on the structure of the given temporal network. In general, since the algorithm explicitly goes through every dynamic graphlet that it counts, the running time is proportional to the number of dynamic graphlets. For a network with $D \Delta t$-adjacent event pairs, counting all dynamic graphlets with up to $k$ events takes $O(|E| + |E|(|E|/|E|)^{k-1})$. As with static
graphlets, the running time of exhaustive dynamic graphlet counting is exponential in graphlet size (but is still practical, as we will show). Yet, as elegant non-exhaustive approaches were proposed for faster static graphlet counting \cite{76, 124, 171}, similar techniques can also be sought for dynamic graphlet counting.

In a network having dense neighborhoods or many events over the same nodes, a given dynamic graphlet type will likely be detected more times than in a network having sparse neighborhoods and few events between the same nodes. Thus, dynamic graphlet counting in the former network type will be computationally expensive, due to having to consider a large number of occurrences of a given graphlet type. Moreover, the occurrences of a given graphlet type will likely be just artifacts of the consecutive snapshots “sharing” the same (dense) network structure. Thus, we propose a modification to the counting process that is expected to reduce the count for a given dynamic graphlet type, as illustrated in Figure 5.3 and described in Appendix Section D.2. This modification, which we call constrained dynamic graphlet counting, is consequently expected to reduce computational complexity of the counting process compared to the regular dynamic graphlet counting procedure described above (which we will demonstrate in Section 5.3). Importantly, this change from regular to constrained dynamic graphlet counting affects only the count of how many times a given dynamic graphlet type appears in the network of interest; it does not affect what graphlet types will be searched for and counted in the network.

5.2.3 Experimental Setup

**Graphlet methods under consideration and network construction.** We compare static, static-temporal, dynamic, and constrained dynamic graphlets. To count static graphlets in a temporal network, we aggregate the temporal data into a single static network, by keeping the node set the same, and by adding an edge between two nodes in the static network if there are at least \( w \) events between these nodes in the
Figure 5.3: Illustration of our dynamic graphlet counting procedure. The temporal network is a sequence of three snapshots. Dashed lines denote instances of the same node in different snapshots. Colored lines denote the path of how the temporal network is explored in order to count the given dynamic graphlet. Regular dynamic graphlet counting will detect all three of the dynamic graphlets $D_1$ (involving nodes $c$ and $f$), $D_2$ (involving nodes $c$, $d$, and $f$), and $D_9$ (involving nodes $a$, $b$, $c$, and $d$). Constrained dynamic graphlet counting (Appendix Section D.2) will detect only the first two dynamic graphlets, but not $D_9$. This is because nodes $c$ and $d$ are interacting in both the second and third snapshot. That is, according to constrained counting, the event between $c$ and $d$ at time $t_3$, which is necessary for identifying a graphlet $D_9$ in the network, is considered to be redundant to the event between $c$ and $d$ at time $t_2$. As such, the event between $c$ and $d$ at time $t_3$ is ignored by constrained counting and thus no $D_9$ can be detected.
temporal network. For other methods, we use a snapshot-based network representation: we split the whole time interval of the temporal network into time windows of size $t_w$, and for each window, we construct a static snapshot by aggregating the temporal data during this window with the parameter $w$, as above. We tested multiple values of $w$ and $t_w$ (Appendix Section D.3). Since we observed no qualitative differences in results of the different choices, we report results for $w = 1$ and $t_w = 2$, unless noted otherwise. Note that in all studied networks all events are instantaneous (i.e., $\sigma_i = 0$ for each event $e_i$). Also, for static and static-temporal graphlets, we vary the number of graphlet nodes $n$, and for dynamic and constrained dynamic graphlets, we vary both the number of graphlet nodes $n$ and the number of graphlet events $k$. Here, we report results for multiple parameter choices.

**Network classification.** An approach that captures network structure (and function) well should be able to group together similar networks (i.e., networks from the same class) and separate dissimilar networks (i.e., networks from different classes) [222]. To evaluate our (constrained) dynamic graphlets against static and static-temporal graphlets in this context, we generate a set of synthetic (random graph) temporal networks of nine different classes corresponding to nine different versions of an established network evolution model (see below) [112]. We use synthetic temporal network data because obtaining real-world temporal network data for multiple different classes and with multiple examples per class is hard. And even if a wealth of temporal network data were available, we typically have no prior knowledge of which real-networks are (dis)similar, i.e., which networks belong to which functional class.

The network evolution model that we use was designed to simulate evolution of real-world social networks. The model is parameterized by the node arrival function that corresponds to the number of nodes in the network at a given time, a parameter that controls the lifetime of a node, and parameters that control how active the nodes are in adding new edges. By choosing different options for the model parameters, we
generate networks with nine different evolution processes (Appendix Section D.3).

To test the robustness of the graphlet methods to the network size, in each of the nine network classes, we test three network sizes: 1000, 2000, and 3000 nodes. For each network size and class, we generate 25 random graph instances. We report results for the largest network size of 3000 nodes. Results for the other network sizes were qualitatively similar.

Given the resulting aggregate or snapshot-based network representations, we then compute static, static-temporal, or (constrained) dynamic graphlet counts in each network and reduce the dimensionality of the networks’ graphlet vectors with principal component analysis (PCA). We consider as few PCA components as needed to account for at least 90% of variation. Here, this leads us to considering the first two PCA components. Then, we use Euclidean distance in this PCA space as a network distance measure and evaluate whether networks from the same class are closer in the graphlet-based PCA space than networks from different classes, as described below.

In addition to studying the nine versions of the above network model that was originally proposed in the domain of social networks, we perform the same analysis on four different versions of two well-established network models from the computational biology domain: geometric gene duplication model with probability cutoff [168] and scale-free gene duplication model [209]. Both models start with a small initial seed network and then grow it by adding new nodes while relying on principles of gene duplication and mutation. The models were shown to mimic well evolution of protein-protein interaction networks [168]. For more details, see Appendix Section D.3.

Node classification. We also evaluate whether the graphlet methods can group together similar nodes (rather than entire networks). We measure the ability of the methods to distinguish between functional node labels (i.e., classes) based on the nodes’ graphlet-based topological signatures. As a proof of concept, we do this on a publicly available Enron dataset [163], which is both temporal and contains node
labels. Unfortunately, availability of additional experimentally inferred temporal and labeled network data is limited. (In Section 5.3.4, we use a computationally inferred network from the computational biology domain to study human aging.) The Enron network is based on email communications of 184 users from 2000 to 2002, with seven user roles in the company as node labels: CEO, president, vice president, director, managing director, manager, and employee.

For an aggregate or snapshot-based network (Appendix Section D.3), we compute static, static-temporal, or (constrained) dynamic graphlet counts of each node in the network and reduce the dimensionality of the given node’s graphlet vector with PCA. Here, we need to keep the first three PCA components to account for at least 90% of variation. We use Euclidean distance as a node distance measure, and evaluate whether same-label nodes are closer in the PCA space than nodes with different labels, as follows.

**Evaluation strategy.** Given a set of objects (networks or nodes), graphlet-based PCA distances between the objects, and the objects’ ground truth classification (with respect to nine/four network classes or seven node labels), we evaluate a graphlet approach by measuring if it correctly places close (far) in the PCA space those objects whose classes match (do not match). First, we sort all object pairs in terms of their increasing distance, and consider $k$ closest pairs. Then, we compute the accuracy in terms of precision, the fraction of class-matching pairs out of the considered pairs, and recall, the fraction of considered class-matching pairs out of all class-matching pairs (Appendix Section D.3). We find the value of $k$ where precision and recall are equal, and we refer to this precision and recall value as the break-even point. Since lower precision means higher recall, and vice versa, we combine the two measures into F-score and report the maximum F-score over all values of $k$. To summarize these results over the whole [0%-100%] range of $k$, we measure average method accuracy by computing the area under the precision-recall curve (AUPR). Also, we compute
the area under the receiver operator characteristic curve (AUROC) (Appendix Section D.3). AUPRs are more credible than AUROCs when there exists imbalance between the size of the set of class-matching object pairs and non-matching pairs. With the expectation that PCA distances between class-matching pairs would be statistically significantly lower than distances of non-matching pairs, we compare two sets of distances via Wilcoxon rank-sum test (Appendix Section D.3) [83]. For each of these evaluation tests, we evaluate all graphlet methods against a random approach (Appendix Section D.3).

5.3 Results and Discussion

We evaluate our novel (constrained) dynamic graphlet approach against the existing static and static-temporal graphlet approaches in the context of two evaluation tasks: network classification (Section 5.3.1) and node classification (Section 5.3.2). Also, we discuss the effect of different method parameters on the results (Section 5.3.3). We present a real-life computational biology application of dynamic graphlets in the context of studying human aging (Section 5.3.4).

5.3.1 Network Classification

First, we test how well the different methods distinguish between nine classes of synthetic temporal networks based on the networks’ graphlet counts; here, the networks come from a well-established network model that was originally proposed in the social network domain. The different evaluation criteria give consistent results: while according to Wilcoxon rank-sum test, all methods have intra-class distances significantly lower than inter-class distances and thus show non-random behavior (p-values less than $10^{-100}$), (constrained) dynamic graphlets are superior in terms of both accuracy and computational complexity, followed by static-temporal and static graphlets, respectively (Figure 5.4 and Appendix Table D.3). Regular dynamic
graphlets perform better than constrained dynamic graphlets in terms of accuracy; the two are comparable in terms of computational complexity (Appendix Tables D.3 and D.4).

Second, we perform the same analysis on four different classes of synthetic temporal networks that come from the computational biology domain. The results are qualitatively similar to those from Figure 5.4: (constrained) dynamic graphlets outperform both static and static-temporal graphlets (Appendix Figure D.1).

5.3.2 Node Classification

Also, we test how well the different methods distinguish between seven different classes of nodes in a real-world network based on the nodes’ graphlet counts. The different evaluation criteria give consistent results: while according to Wilcoxon rank-sum test, all methods have intra-class distances significantly lower than inter-class distances and thus show non-random behavior ($p$-values less than $10^{-100}$), just as with network classification, (constrained) dynamic graphlets are again superior both in terms of accuracy and computational complexity, followed by static-temporal graphlets, followed by static graphlets (Figure 5.4 and Appendix Table D.5).

Constrained dynamic graphlet counting takes significantly less time than regular dynamic graphlet counting (Appendix Table D.5), which justifies our motivation behind constrained counting. This speedup allows us to consider larger graphlet sizes (e.g., six or seven nodes) that are not attainable when using regular dynamic graphlet counting due to computational constraints. Nonetheless, constrained dynamic graphlet counting outperforms regular dynamic graphlet counting even when controlling for graphlet size (Appendix Section D.4 and Appendix Table D.5).

Importantly, the four graphlet methods differ not only quantitatively (as shown above) but also qualitatively: static, static-temporal, and (constrained) dynamic graphlets identify different nodes as topologically similar (Appendix Figure D.2).
Figure 5.4: Comparison of the graphlet approaches in the context of network and node classification, in terms of (a) AUPR and AUROC values and (b) precision-recall curves. For each method, the highest-scoring graphlet size is chosen. For other parameter choices, see Appendix Tables D.3 and D.5.
5.3.3 Effect of Graphlet Size on Results

We test the effect on results of graphlet size in terms of the number of nodes as well as events (Appendix Section D.4). For network classification, increasing the values of these parameters does not always improve accuracy, while it (drastically) increases the running time of graphlet counting (Appendix Tables D.3 and D.4). Thus, using smaller graphlets should suffice to achieve satisfactory accuracy at a reasonable computational complexity. On the other hand, for node classification, using larger graphlets generally improves the performance. Nonetheless, there is an effect of diminishing returns with increase of graphlet size (Appendix Table D.5). This again implies that using smaller graphlets might suffice.

The decrease of performance with increase of graphlet size in the task of network classification (but not node classification) is not alarming. Similar behavior has already been observed in static network research [84, 222]. A possible explanation for such behavior is discussed in Appendix Section D.4. Further theoretical and empirical analyses of this behavior are subject of future work.

5.3.4 Application to Aging

Motivation. As susceptibility to diseases increases with age, studying human aging is important. However, doing so experimentally is hard due to long lifespan and ethical constraints. Network research can be used to deepen current limited knowledge about human aging that has been obtained mainly via other computational methods, e.g., analyses of gene expression or sequence data [99]. Here, we aim to complement existing static [54, 59] or static-temporal [54] network efforts to study human aging with our new temporal approach – (constrained) dynamic graphlets.

We already used a static graphlet-based node centrality measure [135], along with six other centrality measures, to study human aging (Section 5.1.2) [54]. Because it is hard to experimentally obtain large-scale temporal molecular network data due
to limitations of biotechnologies for data collection, we integrated the current static protein-protein interaction network of human \cite{158} with aging-related gene expression data \cite{19} to computationally infer temporal age-specific network data. Then, we predicted as aging-related those genes whose network centralities significantly changed with age. In that study, we computed centrality of each node in each snapshot and analyzed the time-series of the results. Hence, that was a static-temporal approach. The study resulted in the set of 537 aging-related gene candidates, which we called DyNetAge, and which we validated in a number of ways.

Here, we use the same temporal age-specific network data and apply our dynamic graphlets to this data set, to see whether we can improve the prediction quality (i.e., gain additional aging-related predictions) compared to the static-temporal DyNetAge approach. Also, since the latter is quite different than our graphlet approaches from this study (as it is based on the notion of changing node centralities and on multiple measures of network topology), we also compare our dynamic graphlets to the static and static-temporal graphlet approaches from the previous sections. This allows for a fair evaluation of the effect on prediction accuracy of the amount of temporal information that each graphlet approach can consider.

**Evaluation on known “ground truth” aging-related data.** Given the temporal network and two mutually exclusive node labels (i.e., classes), corresponding to “aging-related gene” if the given node is present in DyNetAge or “non-aging-related gene” if the given node is absent from DyNetAge, we perform the same node classification analysis as in Section 5.3.2: we ask whether aging-related genes are closer to each other than to non-aging-related genes in the graphlet-based PCA space (here, we need to consider the first two PCA components to account for at least 90% of variation; Section 5.2.3). We find that the results are consistent with those from Section 5.3.2 (constrained) dynamic graphlets are again superior compared to static and static-temporal graphlets (Appendix Figure D.3). Moreover, similar results hold
even when we use non-network-based “ground truth” aging-related data instead of DyNetAge (Appendix Figures D.4 and D.5).

The above analysis mimics precisely our node classification analysis from Section 5.3.2. Next, we perform a modified analysis: we do not consider the entire $k$ range (Section 5.2.3) but instead focus only on high-scoring node pairs, i.e., on a low $k$ threshold. We do this because we are interested in making as accurate predictions as possible (corresponding to higher precision) at the expense of reducing the number of predictions (corresponding to lower recall). Since precision drastically decreases as we increase $k$ while recall barely increases (Appendix Figure D.3), we study only the highest-scoring node pairs. For illustration purposes, we choose two such values of $k$: 0.00005% and 0.0001%. Since we are dealing with millions of node pairs, even such low $k$ values result in sufficiently many (hundreds of) high-scoring node pairs. When we compare precision of the graphlet methods at each of the two values of $k$, our dynamic graphlets are again comparable or superior to static and static-temporal graphlets (left side of Table 5.1). All methods have the same (low) recall of 0.001, as expected for such small values of $k$. Similar results hold when we use non-network-based “ground truth” aging-related data (Appendix Table D.6).

Finally, we perform an additional analysis to evaluate the different graphlet approaches in the context of aging. Since in the following subsection we aim to predict novel aging-related knowledge, and since there are significantly many more non-aging-related than aging-related genes in the network data, we perform the same evaluation as above (considering the same two $k$ values) while discarding all of the highest scoring node pairs in which both genes are non-aging-related. We do this because node pairs in which both genes are non-aging-related would mistakenly lead to high precision (this is exactly why the random approach has a high precision in the left side of Table 5.1). Again, we find that dynamic graphlets are overall superior to static and static-temporal graphlets (right side of Table 5.1). Note that the higher precision
values in the previous analysis (left side of Table 5.1) compared to this analysis (right side of Table 5.1) are expected. This is because the number of false positives (gene pairs involving an aging-related gene and a non-aging-related gene) stays the same, while the number of true positives decreases (since we have removed some of the true positive gene pairs, i.e., those pairs involving two non-aging-related genes).

**Evaluation of novel aging-related predictions.** The above analysis suggests that because dynamic graphlets do not achieve precision of 1 (Table 5.1), they rank as high-scoring a node pair in which one gene is aging-related while the other one is not. Given this scenario, we hypothesize that the non-aging-related gene in such a pair is actually aging-related (i.e., that it was missed by the DyNetAge study) and we predict it as such. We generalize this prediction strategy to all graphlet approaches. This way, static, static-temporal, dynamic, and constrained dynamic graphlets produce 84, 16, 16, and 80 novel aging-related predictions at the first \( k \) threshold and 86, 43, 47, and 81 novel predictions at the second \( k \) threshold, respectively (Appendix Figure D.6 and Appendix Table D.7).

Next, we aim to validate each approach’s predictions by measuring their gene, functional (i.e., Gene Ontology (GO)) [9], and disease [47] overlaps with independent known “ground truth” aging-related data, as described in [54]. The more statistically significant and larger the overlap, the better the given approach. Here, we find that dynamic or constrained dynamic graphlets are superior to static and static-temporal graphlets in 67% of all tests, and they are comparable in the remaining 33% of all tests (Table 5.2). Thus, (constrained) dynamic graphlets not only improve the prediction quality of the DyNetAge approach and uncover the known aging-related knowledge more precisely than static or static-temporal graphlets, but also, more of their novel knowledge can be validated.

When we qualitatively zoom into the results from Table 5.2, we find evidence that
TABLE 5.1

PRECISION OF THE DIFFERENT METHODS IN THE CONTEXT OF AGING AT THE TWO k VALUES WHEN CONSIDERING ALL NODE PAIRS (LEFT) AND IGNORING ALL NODE PAIRS IN WHICH BOTH GENES ARE NON-AGING-RELATED (RIGHT)

<table>
<thead>
<tr>
<th>Method</th>
<th>Considering all node pairs</th>
<th>Ignoring non-aging-related node pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k₁</td>
<td>k₂</td>
</tr>
<tr>
<td>Static</td>
<td>0.981</td>
<td>0.981</td>
</tr>
<tr>
<td>Static-temporal</td>
<td>0.992</td>
<td>0.992</td>
</tr>
<tr>
<td>Dynamic</td>
<td>0.998</td>
<td>0.998</td>
</tr>
<tr>
<td>Constrained dynamic</td>
<td>0.993</td>
<td>0.993</td>
</tr>
<tr>
<td>Random</td>
<td>0.850</td>
<td>0.851</td>
</tr>
</tbody>
</table>

For each method, the highest-scoring graphlet size is chosen. In a column, the value in bold is the best result over all methods.

the novel knowledge predicted by (constrained) dynamic graphlets is indeed aging-related. First, functional overlap is significant ($p$-values below 0.05) between dynamic graphlets and each “ground truth” data set, for each $k$ value; only in one case, $p$-value is only marginally significant (0.06). Some significant functional overlaps also exist for static-temporal graphlets. For the second $k$ value, only dynamic graphlets significantly overlap ($p$-value of $4 \times 10^{-4}$) with SequenceAge [44], the most trusted source of “ground-truth” aging-related knowledge in human obtained mainly via genomic sequence analyses. Second, for both $k$ values, only predictions resulting from (constrained) dynamic graphlets are enriched in some diseases that the “ground truth” data are also enriched in; this does not hold for static or static-temporal graphlets. Dynamic graphlets’ disease overlap with SequenceAge is significant ($p$-value of 0.03) for the second $k$ value.

Zooming further into the two overlaps of DyNetAge and SequenceAge at the sec-
### TABLE 5.2

OVERLAPS (GIVEN AS PERCENTAGES OF THE SMALLER OF THE COMPARED DATA SETS), ALONG WITH $p$-VALUES OF THE OVERLAPS (SHOWN IN PARENTHESES), OF: 1) GENES, 2) ENRICHED FUNCTIONS, AND 3) ENRICHED DISEASES, BETWEEN EACH GRAPHET APPROACH’S NOVEL PREDICTIONS AND THE THREE INDEPENDENT “GROUND TRUTH” AGING-RELATED DATA SETS (BrainExpression2004Age, BrainExpression2008Age, AND SequenceAge ), FOR THE TWO $k$ VALUES

<table>
<thead>
<tr>
<th>Overlap type</th>
<th>Method</th>
<th>“Ground-truth” aging-related data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BrainExpression2004Age</td>
</tr>
<tr>
<td></td>
<td>$k_1$</td>
<td>$k_2$</td>
</tr>
<tr>
<td>1. Gene</td>
<td>Static</td>
<td>2.4% (0.81)</td>
</tr>
<tr>
<td></td>
<td>Static-temporal</td>
<td>0.0% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>0.0% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Constrained dynamic</td>
<td>3.8% (0.56)</td>
</tr>
<tr>
<td>2. Functional</td>
<td>Static</td>
<td>9.1% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Static-temporal</td>
<td>10.5% (0.05)</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>10.5% (0.05)</td>
</tr>
<tr>
<td></td>
<td>Constrained dynamic</td>
<td>2.9% (N/A)</td>
</tr>
<tr>
<td>3. Disease</td>
<td>Static</td>
<td>0.0% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Static-temporal</td>
<td>0.0% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>0.0% (N/A)</td>
</tr>
<tr>
<td></td>
<td>Constrained dynamic</td>
<td>14.3% (N/A)</td>
</tr>
</tbody>
</table>

*The first two “ground truth” data sets have been derived via gene expression analyses, whereas the latter has been derived via genomic sequence analyses; see [54] for details. “N/A” is shown when there are fewer than two objects (genes, functions, or diseases) in the overlap. Statistically significant $p$-values (at 0.05 threshold) are shown in bold. Note that low $p$-values are extremely encouraging, since we are aiming to validate novel aging-related knowledge. For the same reason, it is not necessarily discouraging when a result is not statistically significant. Also, note that a larger relative (percent) overlap between two sets does not necessarily mean a lower $p$-value, as the $p$-value also depends on the size of the two sets of interest; see the description of the hypergeometric test in e.g., [54]. There are $3 \times 3 \times 2 = 18$ combinations of overlap type, “ground truth” data, and value of $k$. For each combination, the value in gray is the best result over the four graphlet methods. By “best result”, we mean the lowest $p$-value, if at least one of the four $p$-values is significant; otherwise, we mean the largest overlap, unless the overlap is 0. In 12/18 = 67\%, 6/18 = 33\%, and 0/18 = 0\% of the combinations, (constrained) dynamic graphlets are superior, comparable, or inferior, respectively, to static and static-temporal graphlets.*
ond value of \( k \), while nine GO terms overlap between dynamic graphlets’ predictions and SequenceAge, eight additional GO terms are enriched in predictions of dynamic graphlets but not in those from SequenceAge (Appendix Table D.8). Similarly, while four diseases (Parkinson disease, oral cancer, chronic obstructive airway disease, and embryoma) overlap between the two data sets, bipolar disorder is also enriched in dynamic graphlets’ predictions but not in SequenceAge. However, we find literature evidence that many of these GO terms and diseases that are enriched in dynamic graphlets’ predictions but not in SequenceAge are actually linked to aging. One such GO term is *response to zinc ion*, and we find that carnosine, which serves as a metal ion (e.g., zinc) chelator, possesses anti-aging functions (PubMed ID (PMID): 25158972). Another such GO term is *rRNA processing*, and we find that human N-acetyltransferase 10 (NAT10) is both a promising target for therapies against premature aging and a catalizer involved in rRNA processing (PMID: 25411247). Additional such GO terms are *xenobiotic metabolic process*, *cellular metabolic process*, *cellular protein metabolic process*, and we find the following. First, when analyzing gene expression profiles associated with the early steps of age-related cell transformation, bioinformatics analyses highlighted metabolic pathways, the top scoring one being the metabolism of xenobiotics by cytochrome P450 (PMID: 24929818). Second, a link has already been established between metabolism and many aging-related diseases, including Alzheimer’s disease, stroke, diabetes (PMID: 25538685). The disease enriched in dynamic graphlets’ predictions but not in SequenceAge is *bipolar disorder*, and we find the following. First, mood disorders, such as bipolar one, are associated with accelerated aging [189]. Second, protein S100B is both elevated in mood disorders and linked to aging, with higher levels in elderly depressed subjects (PMID: 23701298). Third, bipolar disorder causes mitochondrial dysfunction [91], and the latter in turn is a cause of aging (PMID: 18226094). These results further validate novel aging-related predictions of dynamic graphlets. Importantly, dynamic
graphlets’ literature-validated connections between GO terms or diseases and aging have been missed by SequenceAge. This implies that our dynamic graphlet approach can be used to complement not just the existing (static and static-temporal) network research but also non-network genomic sequence research.

5.3.5 Implementation of the Methods

We make publicly available our software for (constrained) dynamic graphlet counting. Given as input a temporal network of interest and the (maximum) graphlet size to be considered, the software outputs a vector of (constrained) dynamic graphlet counts for the entire temporal network as well as for each individual node. To obtain static or static-temporal graphlet counts of the entire aggregated network or of individual temporal snapshots, respectively, one can use GraphCrunch [103, 131].

As different input parameter values may be optimal for different network types (e.g., networks from different domains), we recommend testing several different combinations, as permitted by computational resources. A good starting point could be testing graphlets with 3-5 nodes and 5-7 events.

5.4 Conclusions

The increasing availability of temporal real-world network data has raised new challenges to network researchers. While one can use the existing static approaches to study the aggregate or snapshot-based network representation of the temporal data, doing so overlooks important temporal information from the data. Hence, we develop a novel approach of dynamic graphlets that can capture the temporal information explicitly. In a systematic and thorough evaluation, we demonstrate the superiority of our approach over its static counterparts. This confirms that efficiently accounting

http://www.nd.edu/~cone/DG
for temporal information helps with structural and functional interpretation of the
network data. This in turn illustrates real-life relevance of our new dynamic graphlet
methodology, especially because the amount of available temporal network data is
expected to continue to grow across many domains, including computational biology.

5.5 Computational Novelty of the Work

We propose novel algorithmic directions for analyzing dynamic network data to
gain insights into the underlying structural changes. We take the graphlet ideas to the
next level to allow for truly dynamic analysis of temporal networks and, as a result,
we introduce the notion of dynamic graphlets. By incorporating both topological
and temporal information, dynamic graphlets help in analyzing the structure and
dynamics of temporal networks. We discuss the ways of generating dynamic graphlets
of arbitrary sizes as well as the methods for their counting. We apply dynamic
graphlets to the problem of both global and local network analysis and demonstrate
that they outperform state-of-the-art static approaches. Further, we use our dynamic
graphlet framework to study age-specific structural and functional changes in the
cell from temporal aging-related molecular network data of human. We have made
publicly available our implementations of the methods and the list of genes predicted
as aging-related by any of the considered methods.2

5.6 My Contributions to the Work

I helped design the study and methods, implemented most of the methods, carried
out most of the computational analyses, helped analyze the results, and helped write
the manuscript.

2http://www.nd.edu/~cone/DG
CHAPTER 6

SCOUT: SIMULTANEOUS TIME SEGMENTATION AND COMMUNITY DETECTION IN DYNAMIC NETWORKS

The work presented in this chapter has been submitted as follows:


6.1 Introduction

6.1.1 Motivation

Networks (or graphs) are elegant yet powerful abstractions for studying complex systems in various domains, from biological entities to social organizations [145]. Real-world systems evolve over time. However, until relatively recently, dynamic measurements about their functioning have been unavailable, owing mostly to limitations of technologies for data collection. Hence, an evolving system has traditionally been analyzed by studying its static network representation, which discards the system’s time dimension by combining all of its interacting elements and their connections across multiple times into a single aggregate network. For example, dynamic cellular functioning has traditionally been modeled as a static protein-protein interaction network that combines biomolecular interactions across different time points and other contexts [20, 130]. However, such an aggregate approach loses important temporal information about the functioning of evolving real-world systems [77]. Analyzing
dynamic network representations of evolving systems is crucial for understanding important mechanisms behind various dynamic phenomena such as human aging in the computational biology domain [54] or opinion formation in the social network domain [174], especially with the increasing recent availability of temporal real-world data in these and other domains. The dynamic network representation of an evolving system models its temporal measurement data as a series of snapshots, each of which is a network that encompasses the temporal data observed during the corresponding time interval. We refer to this snapshot-based representation as a dynamic network.

Approaches for studying dynamic networks can be categorized into: 1) those that extend well-established static network problem formulations and solutions to their dynamic counterparts, and 2) those that consider novel network problems and solutions that arise specifically from the time dimension and are thus native only to the dynamic setting. A popular problem from category 1 above that is of our interest is community detection. A popular problem from category 2 above that is of our interest is time segmentation, or segment detection (also known as change detection). We next discuss these two problems.

Community detection aims to study network structure (or topology) from mesoscopic (i.e., intermediate or groups-of-nodes level) perspective, in contrast to doing so from macroscopic (i.e., global or network level) or microscopic (i.e., local or node level) perspective [61]. Specifically, the goal of community detection is to identify groups of topologically related (e.g., densely interconnected [149, 221] or topologically similar [85, 130, 194]) nodes called communities (or clusters), which are likely to indicate important functional units within the network. For example, communities can correspond to proteins with similar functions in a biological network or groups of friends in a social network [61, 79, 216]. A partition is a division of a network into communities, with each node belonging to a single community. We focus on this mathematical notion of a partition; that is, we consider non-overlapping com-
munities. Nonetheless, our work can be extended to handle overlapping communities as well. For an evolving real-world system, community detection in its dynamic network representation is likely to yield additional insights compared to community detection in the system’s static network representation [72, 129]. Two extremes of community detection in a dynamic network are: 1) snapshot clustering and 2) consensus clustering. On the one hand, snapshot clustering finds a separate partition for each temporal snapshot [32, 65, 94, 143]. Given the snapshot-level partitions, one can then track their evolution by matching individual clusters in adjacent snapshots [21, 69, 154, 176]. On the other hand, consensus clustering finds a single partition that fits well all snapshots [11, 106, 155, 229]. In the real life, community organization most often lies between these two extremes. Finding this real life community organization is one of key goals of our study.

Segment detection aims to divide a dynamic network into continuous segments (groups of snapshots), such that the “border” between each pair of adjacent segments marks a prominent shift in the network structure [172]. As a result, all snapshots within a given segment have similar network structure, while every two adjacent segments have snapshots with dissimilar structure. The set of all segments covering the whole dynamic network is called the segmentation of the network. Time points that separate the segments are called change points. Since change points correspond to shifts in the network structure, they likely indicate functionally important events in the life of the underlying system [172]. For example, change points can correspond to transitions between different functional states in brain networks or to stock market changes in financial networks [15]. Finding change points indicating important structural shifts in the dynamic network is the other key goal of our study.

There is a connection between community detection and segment detection. The former aims to partition a dynamic network along the node dimension (by grouping nodes into communities), while the latter does this along the time dimension (by
grouping snapshots into segments). The combination of the two problems, which is our focus and which we refer to as segment community detection (SCD), can be seen as two-dimensional clustering: simultaneously grouping snapshots of the dynamic network into segments based on community organization of the snapshots, and grouping nodes of the snapshots into communities based on the segments these snapshots belong to (Figure 6.1).

SCD naturally allows for achieving the goal of compromising between the two extremes of snapshot clustering and consensus clustering to identify the real life community organization. Namely, while snapshot clustering is set to “zoom-in” to the level of individual snapshots and consensus clustering is set to “zoom-out” to the level of the whole dynamic network, SCD allows for automatically choosing an appropriate “zoom level” by focusing on segments, each potentially spanning multiple coherent snapshots while still capturing important changes in the community organization (Figure 6.2). As an illustration, consider studying how protein modules evolve with age: it might be more desirable to focus on different stages of the aging process such as infancy, childhood, adolescence, adulthood, etc. \[138\] (via SCD) than on each
Figure 6.2: Illustration of how SCD (middle) naturally allows for compromising between the two extremes of snapshot clustering (bottom) and consensus clustering (top), by balancing the number of segments (which we refer to as segmentation parsimony; Section 6.2.2) and how well a partition fits the corresponding snapshot (which we refer to as partition accuracy; Section 6.2.2). At each of the three horizontal levels, in blue we show the same community across different snapshots within the given segment.

day/month/year of the lifespan (via snapshot clustering) or on the entire lifespan (via consensus clustering). Similar holds when studying evolution of protein modules with disease (e.g., cancer) progression.

6.1.2 Related Work

Several approaches exist that can be adapted to be able to deal with the SCD problem: GraphScope [48, 58, 199], Multi-Step [11], and GHRG [156]. For a review of how these methods work, see Appendix Section E.1. These existing methods can produce both segments and their corresponding partitions, which indeed is a solution that SCD aims to find. However, these approaches have the following drawbacks. 1) They generally cannot produce a high-quality solution with respect to both of the SCD aspects (i.e., segmentation quality and partition quality), as we will show in Section 6.3. 2) For each method, either: a) the number of segments can only be determined automatically but not set by the user, or instead b) the number of segments
can only be set by the user but not determined automatically. In applications where some domain expert knowledge on the desired number of segments is available, the user should be able to feed this knowledge into the method by setting the number of segments, but the methods of type “a” above (GraphScope and GHRG) cannot handle this. On the other hand, in applications where such knowledge is unavailable, the method should be able to determine an appropriate number of segments automatically, but the methods of type “b” above (Multi-Step) cannot handle this. For a method to be generalizable to both types of applications, the method should be able to handle both automatic as well as user-defined determination of the appropriate number of segments. 3) Each of the existing methods has a single built-in intuition about what a good segment or partition is. Hence, each approach could be biased towards the particular parameters that it implements. Thus, a more generalizable approach that would offer flexibility in terms of parameter choices is desirable. To address these three drawbacks, we introduce SCOUT, a new general framework for segment community detection, as follows.

6.1.3 Our Approach and Contributions

We propose a novel formulation of the SCD problem as an optimization process that integrates the two aspects (segment detection and community detection) more explicitly than the existing methods. Also, we propose SCOUT, a general framework for solving the new problem, which addresses the drawbacks of the existing methods: 1) it is capable of producing a high-quality solution with respect to both of the SCD aspects; 2) it can handle both automatic and user-defined determination of the appropriate number of segments; 3) it offers high level of flexibility when it comes to the choice of segmentation or partition quality parameters.

Specifically, SCOUT algorithm consists of three key parts: objective function (a measure of what a good SCD solution is), consensus clustering (given a set of change
points, how to find a good partition for each segment), and search strategy (how to search through the space of possible change point sets). We vary choices for each of these three components. By doing so, we effectively trade-off between different goals, such as between segmentation quality and partition quality, or between accuracy and speed.

We comprehensively evaluate SCOUT against the existing methods. We do so on both synthetic and real-world networks of varying sizes. In particular, because in some domains (such as computational biology) large-scale experimental real-world dynamic network data are not available \cite{20, 51, 140, 224}, in order to illustrate generalizability of our approach, we first perform evaluation on synthetic dynamic network data. For this purpose, we introduce an intuitive model for automatic generation of a synthetic dynamic network of an arbitrary size with known ground truth segmentation as well as community organization, and we perform our evaluation on 20 different synthetic ground truth configurations. In addition, we analyze six real-world dynamic networks from domains that do offer such data and that offer such data with some ground truth knowledge embedded into them; these networks span studies of human proximity, communication, and political relationships. To evaluate how well each method can reconstruct the ground truth knowledge, we rely on established partition quality and similarity measures as a basis for developing new SCD accuracy measures that can simultaneously account for both segmentation quality and partition quality. Interestingly, although the existing approaches can all achieve the same task of SCD, they have not been evaluated against each other to date. Hence, our study provides the first ever such evaluation. Importantly, we show that SCOUT overall outperforms the existing methods with respect to both segmentation quality and partition quality, while also being more computationally efficient.
6.2 Methods

6.2.1 Notations

A **dynamic network** $D$ is a sequence of $k$ snapshots $\{G_0, G_1, \ldots, G_{k-1}\}$, where each snapshot $G_i = (V_i, E_i)$ is a static graph capturing network structure during time interval $i$. A sequence of consecutive snapshots can be grouped into a segment. Formally, a **segment** $s$ is a sequence of consecutive snapshots $\{G_i, G_{i+1}, \ldots, G_j\}$, $i \leq j$, with $i$ being its **start time**, $j$ being its **end time**, and $j - i + 1$ being its **length**. A sequence of non-overlapping segments (meaning that each segment in the sequence starts right after the previous one ends) that covers the whole dynamic network (meaning that the first segment in the sequence starts at time 0 and the last segment in the sequence ends at time $k - 1$) forms a segmentation of this network. Formally, a **segmentation** $S$ is a sequence of $l$ adjacent segments $\{s_0, s_1, \ldots, s_{l-1}\}$ such that $s_0$ starts at time 0 and $s_{l-1}$ ends at time $k - 1$. We can specify such a segmentation via a set $T = \{t_1, t_2, \ldots, t_{l-1}\}$ of $l - 1$ time points called **change points**, such that $t_i$ is the start time of segment $s_i$, $i \in [0, l - 1]$ (by convention, we always assume that $t_0 = 0$).

6.2.2 Problem Formulation

Given a dynamic network $D$, the goal of SCD is to simultaneously find a segmentation $S^* = \{s_0^*, s_1^*, \ldots, s_{l-1}^*\}$ (or equivalently a change point set $T^* = \{t_1^*, t_2^*, \ldots, t_{l-1}^*\}$) and a sequence of partitions $P^* = \{p_0^*, p_1^*, \ldots, p_{l-1}^*\}$ such that $S^*$ identifies important shifts in the community organization of $D$ and each $p_i^*$ (called **segment partition**) reflects well the community organization of each snapshot within segment $s_i^*$ (Figure [6.1]). Clearly, the output (i.e., the solution) of the SCD problem can be represented as $O^* = (T^*, P^*)$. Intuitively, in a good output, $T^*$ should be **parsimonious** (meaning that it should capture all important shifts in the network with as small as possible number of change points), while $P^*$ should be **accurate** (meaning that...
segment partitions should correctly capture community organization of all snapshots within the corresponding segment). That is, output $O^* = (T^*, P^*)$ should aim to simultaneously satisfy two objectives: segmentation parsimony and partition accuracy.

We can now state the problem:

**Problem 1 (SCD)** Given a dynamic network $D = \{G_1, G_2, \ldots, G_k\}$, find a number of segments $l$, a sequence of $l - 1$ change points $T^* = \{t^*_1, t^*_2, \ldots, t^*_{l-1}\}$, and a sequence of $l$ segment partitions $P^* = \{p^*_1, p^*_2, \ldots, p^*_l\}$ such that the output $O^* = (T^*, P^*)$ forms a parsimonious segmentation with accurate segment partitions.

In some sense, the two objectives, segmentation parsimony and partition accuracy, are competing with each other. That is, optimizing one does not necessarily lead to optimizing the other. For example, at the extreme of snapshot community detection (bottom of Figure 6.2), each snapshot is considered to be a separate segment that has its own well-fitting partition, which yields high partition accuracy. However, such a fine-grained output with the maximum possible number of segments might contain redundancies, because some adjacent snapshots might have similar community organizations. In this case, segmentation parsimony will be low. To optimize (increase) segmentation parsimony, adjacent snapshots with similar community organizations should be grouped together. At the other extreme of consensus community detection (top of Figure 6.2), all snapshots are grouped together into one segment with a single common segment partition for the whole network, which yields high segmentation parsimony. However, the single segment partition will have to “compromise” between many possibly quite distinct snapshots. In this case, the segment partition will not be able to fit well all of the distinct snapshots, and consequently, partition accuracy will be low. In real-world scenarios, the SCD solution typically lies between these two extremes, but finding such a solution still requires balancing between the two somewhat contradicting goals of optimizing both segment parsimony and partition accuracy. We formalize the ways of finding such a solution in Section 6.2.3.
Algorithm 6.1 SCOUT overview

Input: \( D \)

Output: \( O^* = (T^*, P^*) \)

1: \( R \leftarrow \emptyset \)
2: \( T \leftarrow \text{SearchStrategy}(\emptyset, R) \)
3: while \( T \neq \emptyset \) do
4: \( S \leftarrow \text{GetSegmentation}(T, D) \)
5: \( P \leftarrow \emptyset \)
6: for \( s \in S \) do
7: \( P \leftarrow P \cup \{\text{ConsensusClustering}(s)\} \)
8: end for
9: \( O \leftarrow (T, P) \)
10: \( R \leftarrow R \cup \{(O, \text{ObjectiveFunction}(O))\} \)
11: \( T \leftarrow \text{SearchStrategy}(T, R) \)
12: end while
13: return \( \text{GetBestOutput}(R) \)

Recall from Section 6.1.2 the need of being able to find a solution with a user-specified number of segments \( l \), in addition to being able to determine this parameter \( l \) automatically. Our current SCD problem formulation (Problem 1) can handle the latter scenario, but we can extend it to handle the former scenario as well. Specifically, when finding an SCD solution, in addition to allowing for simultaneously optimizing both aspects of SCD quality (i.e., segmentation parsimony and partition accuracy), we can allow for optimizing only one aspect (partition accuracy) while setting the other one (segmentation parsimony, expressed as the number of segments \( l \)) as a constraint. So, we extend the problem formulation by adding to the existing SCD objective from Problem 1 the following new objective: given a dynamic network \( D \) and the desired number of segments \( l \) as input by the user, find an output \( O^* = (T^*, P^*) \) with \( l \) segments that achieves the highest partition accuracy. We refer to this new objective as the constrained SCD problem (CSCD). We propose SCOUT to solve any of the SCD and CSCD problems, in order to allow for handling both of the above scenarios (automatic vs. user-defined selection of the number of segments \( l \), respectively), as follows.
6.2.3 Our SCOUT Approach

Given a dynamic network $D$, we aim to find an output $O^* = (T^*, P^*)$ by directly optimizing an objective function that measures both segmentation parsimony and partition accuracy (see below for details on how we deal with SCD versus CSCD). Algorithm 6.1 provides a high-level overview of SCOUT, and Appendix Section E.2 and Appendix Figures E.1-E.3 provide further details. SCOUT has the following five steps.  

1) Select the initial change point set as the current change point set $T$ (line 2 in Algorithm 6.1). For example, the initial change point set could correspond to a set of all possible snapshot-level segments (bottom-up search) or just one large network-level segment (top-down search). Given $T$, the method iteratively performs the following steps.  

2) Perform consensus clustering within each segment $s_i$ to get its corresponding partition $p_i$ (line 7). In general, the consensus clustering method should aim to obtain the partition set $P$ that maximizes the objective function for $T$. Step 2 results in $O = (T, P)$ (line 9).  

3) Use a search strategy to search for the next change point set that will become the new current change point set $T$ (line 11). Clearly, the search strategy guides how we explore the space of possible change point sets. For example, in bottom-up search, the next change point set is obtained by merging two adjacent segments, while in top-down search, the next change point set is obtained by splitting a segment into two.  

4) Repeat steps 2 and 3 above until the exploration of the space is finished (corresponding to $T = \emptyset$ in line 3), e.g., until one largest possible network-level segment is reached in bottom-up search or until all possible snapshot-level segments are reached in top-down search.  

5) Choose the best output out of all outputs computed in step 2 as the final output $O^* = (T^*, P^*)$ (line 13). When solving the SCD problem, the best output is the one maximizing the objective function. When solving the CSCD problem, the best output is the one maximizing the objective function while satisfying the constraint (the solution consisting of $l$ segments). Thus, SCOUT contains three main components: objective
function (Appendix Section E.2.1), consensus clustering (Appendix Section E.2.2), and search strategy (Appendix Section E.2.3).

6.2.4 Experimental Setup

6.2.4.1 Methods for Comparison

We compare SCOUT against the three existing approaches: GraphScope, Multi-Step, and GHRG. We discuss the methods’ parameters that we use in Appendix Section E.3.1.

6.2.4.2 Datasets

We evaluate the methods on two types of networks: synthetic networks and real-world networks.

**Synthetic networks.** To generate a synthetic dynamic network $D$ with the embedded ground truth $O^{(gt)} = (T^{(gt)}, P^{(gt)})$, we introduce a new dynamic random graph model for this purpose, which we call segment community generator (SCG), and which works as follows. We assume that the following are provided as input by the user: the number of snapshots $k$, the number of segments $l$, the number of nodes in each snapshot $n$, the minimum required number of nodes in each cluster $c_{min}$, and two parameters $c_{in}$ and $c_{out}$ that control intra- and inter-community edge density of the snapshots. The process of generating a synthetic dynamic network with these parameters contains four steps (Appendix Figure E.4). In the first three steps, we generate the ground truth $O^{(gt)} = (T^{(gt)}, P^{(gt)})$, and in the last step, we use $O^{(gt)}$ to actually generate snapshots of $D$. Intuitively, we: 1) generate the set of change points $T^{(gt)}$ to define segments, 2) create a special auxiliary graph describing how segment partitions evolve from segment to segment, 3) use this graph to generate the actual segment partitions $P^{(gt)}$, and 4) use a stochastic blockmodel to generate snapshots of $D$, based on the idea that snapshots within the same segment (as defined
by $T^{(gt)}$ have the same community organization (as defined by the corresponding segment partition from $P^{(gt)}$). For details on each step, see Appendix Section [E.3.2] and Appendix Algorithm [E.1].

For our experiments, we generate synthetic dynamic networks with 16 snapshots and 1, 2, 4, 8, and 16 ground truth segments. We also consider networks of various sizes: 50, 100, 500, and 1000 nodes in each snapshot. This results in $5 \times 4 = 20$ different synthetic network configurations. In each configuration, we set the parameters as follows. For partition graph $G_P$, we set $c_{min} = 5$ when $n = 50$, $c_{min} = 10$ when $n = 100$ and $c_{min} = 50$ when $n \in \{500,1000\}$. For the stochastic blockmodel, we set $c_{in} = 32$ and $c_{out} = 8$. For each synthetic network configuration, we generate 10 random instances in order to account for the randomness in the synthetic network generator. This totals to $20 \times 10 = 200$ synthetic networks.

**Real-world networks.** Unlike our synthetic networks, real-world networks that we analyze (see below) do not contain the ground truth in the form of $O^{(gt)} = (T^{(gt)}, P^{(gt)})$. The only appropriate ground truth knowledge that we have and that we have only for some of the networks is the set of change points $T^{(gt)}$. None of the networks contain the set of segment partitions $P^{(gt)}$ as the ground truth, either because they do not have available any node community structure information whatsoever or because they only have available an inappropriate single static community structure for the whole dynamic network. Nevertheless, we can still evaluate the methods on the real-world networks, by: 1) using evaluation measures that do not rely on the ground truth knowledge, for all real-world networks, and 2) assessing how well the methods can recover the change point set $T^{(gt)}$, for real-world networks that do contain this ground truth knowledge.

We consider six different publicly available real-world dynamic networks. 1) **Hypertext** [86] network contains information about face-to-face proximity of attendees of the Hypertext 2009 conference. The nodes correspond to people, and there is an edge
between two people if they were close to each other within a given time interval, as measured by wearable radio badges. This network has $T^{(gt)}$ that corresponds to the list of events from the conference program [86]. 2) AMD Hope [204] network contains information about co-location of attendees of The Last HOPE conference in 2008. The nodes correspond to people, and there is an edge between two people if they were located in the same room at the same time. This network has $T^{(gt)}$ that corresponds to the featured/keynote talks and social events [204]. 3) High School [125] network contains information about proximity of students in a high school during one work week in 2013. The nodes and edges are added in the same way as in Hypertext network. This network does not have $T^{(gt)}$. 4) Reality Mining [19] network contains information about social interactions of university students and faculty during 2004-2005 academic year. The nodes correspond to people, and there is an edge between two people if there was a phone call between them in a given time interval. This network has $T^{(gt)}$ that corresponds to the list of events from the academic calendar [156]. 5) Enron [163] network contains information about email communication of employees of the Enron corporation during the 2000-2002 period. The nodes correspond to people, and there is an edge between two people if there was an email between them in a given time interval. This network has $T^{(gt)}$ that corresponds to the list of company-related events from the news sources [156]. 6) Senate [94] network contains information about voting similarities of United States senators during the 1789-2015 period (i.e., for 113 Congresses). The nodes correspond to states, and there is an edge between two states if the voting similarity between the corresponding senators in a given time interval is high enough [94]. This network does not have $T^{(gt)}$. This is because for Senate network we cannot use the list of historic events as a formal ground truth change point set, since it is not clear how to objectively select a fixed number of them (i.e., how to determine which events are more important than others and how many of the most important events should be considered). For statistics of
6.2.4.3 Evaluation Measures

We evaluate the performance of a given method via: network structure-based measures and ground truth knowledge-based measures.

Network structure-based measures. Here, we measure the quality of the results of a given method with respect to the structure of the input dynamic network $D$, without relying on any ground truth knowledge. Specifically, we can use one of the objective functions from Appendix Section E.2.1 to measure the quality of the method’s $O^* = (T^*, P^*)$. These objective functions include four $Q_P$ measures of partition quality and two $Q_B$ measures accounting for both segmentation quality and partition quality. Regarding the four $Q_P$ measures (i.e., modularity, conductance, normalized cut, and average-ODF), in our experiments, all four measures show statistically significantly correlated results with respect to both Pearson and Spearman correlations (with all pairwise $p$-values $< 10^{-49}$). So, in case of $Q_P$, for brevity, we report results only for modularity. Regarding the two $Q_B$ measures (i.e., AIC and BIC), we do not evaluate the results with respect to them, since these are the objective functions that SCOUT explicitly aims to optimize, and thus, we want to avoid circular reasoning.

Ground truth knowledge-based measures. Here, we measure the quality of the results of a given method with respect to the available ground truth knowledge. We discuss two general ways to achieve this: I) by measuring similarity of the method’s $O^* = (T^*, P^*)$ to the known ground truth $O^{(gt)} = (T^{(gt)}, P^{(gt)})$ and II) by evaluating the method’s ability to rank time points according to how “change point-like” they are.

I) We introduce three general groups of measures of similarity between $O^*$ and $O^{(gt)}$: a) segmentation similarity $\text{Sim}_T$, focusing only on the segmentation aspect of $O^*$ and
a) To measure $\text{Sim}_T$ between $O^*$ and $O^{(gt)}$, intuitively, we first construct for each of them a special time point partition $\mathcal{P}_T$ that captures how the snapshots of $D$ are grouped into segments. For example, for $O^*$ and $O^{(gt)}$ in Figure 6.3, $\mathcal{P}_T^*$ contains three clusters ($\{0, 1, 2, 3\}$, $\{4, 5, 6\}$, and $\{7, 8, 9\}$) and $\mathcal{P}_{T}^{(gt)}$ contains four clusters ($\{0, 1\}$, $\{2, 3, 4, 5\}$, $\{6, 7\}$, and $\{8, 9\}$). Then, we measure similarity between the two resulting time point partitions via an existing partition similarity measure (see below after part “c”). For formal details regarding $\text{Sim}_T$, see Appendix Section E.3.3.

b) To measure $\text{Sim}_P$ between $O^*$ and $O^{(gt)}$, intuitively, we first measure for each snapshot of $D$ similarity between its corresponding segment partitions in $O^*$ and $O^{(gt)}$ via an existing partition similarity measure (see below after part “c”). For example, for $O^*$ and $O^{(gt)}$ in Figure 6.3, for snapshot $G_0$, we measure similarity between $p_0^*$ and $p_0^{(gt)}$ (since $G_0$ belongs to the first segment in $O^*$ and to the first segment in $O^{(gt)}$), while for snapshot $G_2$, we measure similarity between $p_0^*$ and $p_1^{(gt)}$ (since $G_2$ belongs to the first segment in $O^*$ and to the second segment in $O^{(gt)}$). Then, we average the results over all snapshots. For formal details regarding $\text{Sim}_P$, see Appendix Section E.3.3.

c) To measure $\text{Sim}_B$ between $O^*$ and $O^{(gt)}$, intuitively, we first construct for each of them a special node-time partition $\mathcal{P}_B$ that simultaneously captures how snapshots are grouped by $T$s and how nodes are grouped by $P$s. For illustrations of node-time partitions of $O^*$ and $O^{(gt)}$, see Figure 6.3. Then, we measure similarity between the two resulting node-time partitions via an existing partition similarity measure (see below). For formal details regarding $\text{Sim}_B$, see Appendix Section E.3.3.

All of $\text{Sim}_T$, $\text{Sim}_P$, and $\text{Sim}_B$ are parameterized with a measure $H$ of similarity between two partitions. We test four popular such measures $H$: 1) Normalized
Figure 6.3: The process of constructing a node-time partition $P_B$ when computing $Sim_B$. Each black circle corresponds to a node at a given time point. Circles on the same horizontal line correspond to a fixed node at different time points. Circles on the same vertical line correspond to different nodes at a fixed time point. Rectangles illustrate clusters in node-time partitions $P_B^*$ and $P_B^{(gt)}$. 
Mutual Information (NMI) [212], 2) Adjusted Mutual Information (AMI) [212], 3) Adjusted Rand Index (ARI) [212], and 4) V-Measure (VM) [175]. For details of the above measures, see Appendix Section E.3.3. In our experiments, all four measures show statistically significantly correlated results with respect to both Pearson and Spearman correlations (with all pairwise p-values $< 10^{-239}$). So, for brevity, we report results only for NMI.

II) Assessing a given method’s ability to detect ground truth change points $T^{(gt)}$ is important in the task of segment detection [172]. One way to achieve this is via $Sim_T$ from above, which directly compares the given method’s change point set $T^\ast$ against $T^{(gt)}$. $Sim_T$ only takes into account time points that were chosen as change points. That is, $Sim_T$ does not consider time points that were not chosen as change points, even though some of these time points may have still been good change point candidates. Namely, when determining which time points should be change points, a method assigns to each time point a score (or rank) according to how “change point-like” the time point is. So, instead of using “binary” information for each time point $t$ as $Sim_T$ does (i.e., either $t \in T^{(gt)}$ or $t \notin T^{(gt)}$), we can make use of the more complete information on ranking of all time points. An example of why this would be useful is as follows. Even if some ground truth change point $t \in T^{(gt)}$ is not (mistakenly) included into $T^\ast$, we still want the method to rank $t$ higher than some other $t' \notin T^{(gt)}$. $Sim_T$ would fail to capture this information, so we use an alternative evaluation metric, as follows.

Having a ranked list of all time points (for details on how we obtain this list for each method, see Appendix Section E.3.3), we measure a given method’s performance with respect to change point classification via three measures: 1) the area under the precision-recall curve (AUPR), 2) the maximum F-score, and 3) the area under the receiver operator characteristic curve (AUROC). For details of the above measures, see Appendix Section E.3.3. In our experiments, all three measures show statistically
significantly correlated results with respect to both Pearson and Spearman correlations (with all pairwise $p$-values $< 10^{-64}$). So, for brevity, we report results only for AUPR.

6.3 Results and Discussion

We compare four different methods (Section 6.2.4.1): three existing methods (GraphScope, Multi-Step, and GHRG; Section 6.1.2) and our new SCOUT approach (Section 6.2.3). We evaluate the methods on synthetic networks as well as real-world networks (Section 6.2.4.2). We evaluate the methods with respect to network structure-based measures and ground truth knowledge-based measures in the task of the SCD problem (Section 6.2.4.3). As a measure of the former type, we use average snapshot partition quality $Q_P$ based on modularity. As a measure of the latter type, we use a) similarity of a method’s output to the ground truth and b) change point classification. For case “a” above, we compute segmentation similarity $Sim_T$, partition similarity $Sim_P$, and overall similarity $Sim_B$. For all of the three similarity measures, we use NMI to measure partition similarity. For case “b” above, we use AUPR. We measure statistical significance of the improvement of SCOUT over the best of the existing approaches (Appendix Section E.3.4).

When we have the complete ground truth information (on both the segmentation aspect and the partition aspect of the SCD problem) available, which is the case for our synthetic networks, we use all of the above measures, but we trust $Sim_B$ the most, since it captures similarity between a given method’s solution and the ground truth solution with respect to both SCD aspects. When we do not have the complete ground truth information (i.e., when we cannot use the two-aspect $Sim_B$), which is the case for our real-world networks, we assess a given method based on the structure-based measure (i.e., $Q_P$ based on modularity) and whichever ground truth knowledge-based measure we can compute based on the partial ground truth information about the
data. Since in our case the available ground truth information is the list of change points, for the latter, we can use any measure that captures the segmentation aspect of the solution quality. Recall that we have two such measures: $Sim_T$ and change point classification (Section 6.2.4.3). Since we demonstrate in Section 6.3.2.1 that the two measures overall yield consistent results on synthetic networks with known ground truth SCD solution, and since per our discussion in Section 6.2.4.3 change point classification is theoretically more meaningful than $Sim_T$ as it accounts for ranking of all time points rather than only for the identified change points, for brevity, we focus only on change point classification for real-world networks.

Below, we first discuss the effect of parameter choices on method performance, in order to choose the best parameter values for each method (Section 6.3.1). Then, we compare the methods on synthetic (Section 6.3.2) and real-world (Section 6.3.3) networks.

6.3.1 The Effect of Method Parameter Choices

We perform all experiments from this section on synthetic networks, since they have the known ground truth knowledge embedded into them (Section 6.2.4.2). In particular, due to high computational complexity of some of the existing methods and a large number of performed tests, in this section, we use the smallest synthetic data with 50 nodes per snapshot. As discussed above, our main criterion for selecting parameters of a given method is overall ground truth similarity $Sim_B$. Note that GraphScope does not accept any user-specified parameters, and thus we leave it out from consideration in this section.

**Multi-Step.** We test the effect on the method’s performance of the similarity threshold parameter $\theta$, which determines when to stop the segment merging process (Appendix Section E.3.1). We find that there is no $\theta$ value that works well for all of the synthetic network configurations with respect to $Sim_B$ (Appendix Figure E.5(a)).
This is mainly because no single $\theta$ value can reliably estimate the ground truth number of segments across the different configurations (Appendix Figure E.5(b)). Thus, Multi-Step can be used to reliably solve only the CSCD problem where the number of segments is provided as input. So, when comparing Multi-Step against other methods in the context of the SCD problem, we instead ask Multi-Step to solve the CSCD problem with the ground truth number of segments given as input. We refer to this modification of Multi-Step as Multi-Step*. This gives Multi-Step an unfair advantage compared to the other methods, but we have to do this in order to include Multi-Step into comparison.

**GHRG.** We test the effect on the method’s performance of windows size $w$ (Appendix Section E.3.1). After varying its values, we observe that the value $w = 4$ generally leads to the highest $\text{Sim}_B$ (Appendix Figure E.6). Thus, we use $w = 4$ for our experiments.

**SCOUT.** We test the effect on the method’s performance of a) the objective function, b) consensus clustering method, and c) search strategy. We choose $Q_B$ based on BIC as the objective function, sum graph with Walktrap as the consensus clustering method, and the bottom-up search as the search strategy, per our discussion in Appendix Section E.4 and Appendix Figures E.7-E.11.

### 6.3.2 Synthetic Networks

We next evaluate the methods (under their best parameter values from Section 6.3.1) on synthetic networks, which have the ground truth SCD solution embedded into them. We consider 20 different synthetic network configurations: five values for the number of segments times four values for the number of nodes per snapshot (Section 6.2.4.2). These configurations span the whole “spectrum” between the extreme cases of snapshot clustering (where the number of ground truth segments corresponds to the number of snapshots) and consensus clustering (where there is
only one ground truth segment corresponding to the whole dynamic network). For each synthetic network configuration, we generate multiple random network instances (Section 6.2.4.2) and report results averaged over the multiple instances.

Recall that the main idea behind our synthetic network generation process (snapshots within the same segment having the same community organization) aligns well with the intuition of each of the considered methods. Thus, we expect all methods to have a fair chance for recovering the ground truth knowledge, with the exception of Multi-Step, which has an unfair advantage over all other methods, per our discussion in Section 6.3.1. Specifically, recall that we provide the ground truth number of segments as input to Multi-Step. This a priori knowledge gives an unfair advantage to Multi-Step compared to all other methods for all configurations, but this advantage is the most pronounced for the extreme configurations with the minimum and maximum possible numbers of ground truth segments (i.e., with one and 16 segments, respectively; Section 6.2.4.2). This is because for these two types of configurations, the knowledge of the ground truth number of segments guarantees that Multi-Step’s solution will have the correct segmentation: given 16 snapshots (which is the size of our synthetic network data), there is only one way to group the 16 snapshots into one segment (the resulting segment will encompass all 16 snapshots) and only one way to group the 16 snapshots into 16 segments (each segment will encompass exactly one of the snapshots). For the other non-extreme configurations, with more than one but less than 16 segments, while knowing the ground truth number of segments still gives an advantage to Multi-Step (meaning that clearly Multi-Step will produce the correct ground truth number of segments, or equivalently, the correct number of change points), it does not necessarily guarantee that Multi-Step will obtain the correct segmentation (i.e., that the identified change points will be correct). This is because for these non-extreme configurations, there are multiple ways to group snapshots into the given number of segments.
For each synthetic network, we know the corresponding ground truth segmentation and segment partitions, so we can fully utilize the available ground truth knowledge-based measures. Below, we start by discussing results when focusing on a single aspect of the SCD problem at a time: first on a segmentation aspect (i.e., $Sim_T$ and change point classification; Section 6.3.2.1) and second on a partition aspect (i.e., $Q_P$ and $Sim_P$; Section 6.3.2.2). Then, we discuss the results with respect to overall ground truth similarity $Sim_B$ (Section 6.3.2.3). Recall that $Sim_B$ is the most reliable measure, since its captures both aspects of the SCD problem. Thus, for $Sim_B$, we also measure the statistical significance of the improvement of SCOUT over the existing methods (Appendix Section E.3.4). Finally, we compare running times of the methods (Section 6.3.2.4).

6.3.2.1 Segmentation Aspect of the Solution Quality

For $Sim_T$, SCOUT is superior to all other methods, as it achieves the highest scores for 90% of all synthetic network configurations, while the other methods are relatively comparable to each other (Appendix Figure E.12(a)). The remaining 10% (i.e., two) of all configurations in which an existing method (in this case, GraphScope) achieves higher scores are configurations with the two largest numbers of nodes per snapshot and with the maximum possible number of segments (Appendix Figure E.13). The fact that GraphScope has higher $Sim_T$ for these configurations is not necessarily surprising, for the following reason. GraphScope generally produces solutions with more segments than the other methods do, frequently overestimating the ground truth number of segments (Appendix Figure E.14). Consequently, since for the configurations with the maximum possible number of segments, the most that GraphScope can overestimate is the maximum number of segments itself (i.e., the correct solution), GraphScope is expected to achieve higher $Sim_T$ than the other methods. Note that when measuring $Sim_T$ for the extreme configurations
Figure 6.4: Representative $\text{Sim}_T$ and $\text{Sim}_P$ scores for synthetic networks with 100 nodes per snapshot (the largest size for which all methods could be run) and four ground truth segments. The results are averaged over all of the corresponding synthetic network instances. Equivalent results for the remaining synthetic network configurations are shown in Appendix Figure E.13.

with the minimum and maximum possible numbers of segments, we exclude Multi-Step from comparison. This is because, per our discussion from Section 6.3.2, we give Multi-Step an unfair advantage by providing it with the ground truth number of segments as input, which for these extreme configurations means a priori knowing the correct segmentation and thus achieving the perfect $\text{Sim}_T$ (Appendix Figure E.13). Interestingly, for the remaining non-extreme configurations, Multi-Step is always outperformed by SCOUT and at least one of the existing methods (Appendix Figure E.12(a)). Therefore, Multi-Step, which knows the ground truth number of segments a priori typically does not yield a high quality segmentation with respect to $\text{Sim}_T$, whereas SCOUT does produce a high quality segmentation (and it typically does so better than the other methods) despite not having this prior knowledge (Figure 6.4 and Appendix Figure E.13). This is further confirmed by the fact that SCOUT can automatically determine the ground truth number of segments more accurately than the existing methods (Appendix Figure E.14).
Figure 6.5: Representative method comparison for synthetic networks with 100 nodes per snapshot with respect to (a) change point classification, (b) $Q_P$, and (c) $Sim_B$. For a given ground truth configuration, the results are averaged overall all of the corresponding synthetic network instances. Note that for panel (a), we exclude from consideration the configurations with the minimum and maximum possible numbers of ground truth segments (i.e., one and 16). We do this because for these configurations, either there are no change points at all (for one segment) or every time point is a change point (for 16 segments), which means that change point classification cannot be performed (Section 6.2.4.3). In panel (b), the dotted lines correspond to the ground truth score. Equivalent results for the remaining synthetic network configurations are shown in Appendix Figure E.15.
For change point classification, SCOUT is superior to all of the existing methods, as it achieves the highest accuracy for 92% of all synthetic network configurations (Appendix Figure E.12(b)). Among the existing methods, GHRG is generally superior, followed by GraphScope and Multi-Step (Figure 6.5(a) and Appendix Figure E.15(a)). In the remaining 8% of all configurations (which is only one configuration in this case – the configuration with 500 nodes and eight segments; Appendix Figure E.12(b)) where SCOUT is not superior, an existing method (in this case, GraphScope) achieves only marginally higher score (Appendix Figure E.15(a)). Overall, the trends with respect to change point classification are similar to those with respect to \( \text{Sim}_T \), which is not surprising, since both measure the same aspect of the SCD problem. Note that for change point classification, Multi-Step does not have the unfair advantage over the other methods, as it does for \( \text{Sim}_T \) above, since its produced time point ranking depends only on the solutions of the CSCD problem (Appendix Section E.3.3).

6.3.2.2 Partition Aspect of the Solution Quality

For \( Q_P \), SCOUT is superior to all other methods, achieving the highest \( Q_P \) for 70% of all synthetic network configurations (Appendix Figure E.16(a)). Among the existing methods, Multi-Step shows the best results, followed by GraphScope and GHRG that are comparable to each other (Figure 6.5(b) and Appendix Figure E.15(b)). Importantly, SCOUT overall outperforms Multi-Step in terms of \( Q_P \) despite the fact that Multi-Step explicitly maximizes modularity (which is the basis of \( Q_P \); Section 6.2.4.3), while the version of SCOUT under consideration does not rely on \( Q_P \) at all (Section 6.3.1). Note that the configurations on which Multi-Step outperforms SCOUT are mostly those with the maximum possible number of ground truth segments (Figure 6.5(b) and Appendix Figure E.15(b)). This is not necessarily surprising, since for these 16-segment configurations, SCOUT can produce a
solution with \textit{at most} 16 segments, while Multi-Step is guaranteed to produce the solution with exactly 16 segments (Section 6.3). That is, intuitively, Multi-Step’s solution will have a separate segment partition for each snapshot, and each of those partitions aims to maximize modularity and consequently $Q_P$. Importantly, for the configurations where Multi-Step outperforms SCOUT, Multi-Step’s $Q_P$-based superiority is \textit{not} necessarily an advantage. This is because Multi-Step achieves higher $Q_P$ scores even compared to $Q_P$ scores of the ground truth solution (Figure 6.5(b) and Appendix Figure E.15(b)). Thus, even if Multi-Step obtains the highest $Q_P$, its partitions might not necessarily be closer to the ground truth than SCOUT’s partitions, as we justify next.

For $Sim_P$, SCOUT is superior to all other methods, as it achieves the highest $Sim_P$ score for 100\% of all synthetic network configurations (Appendix Figure E.16(b)). The other methods are relatively comparable to each other, with slight superiority of Multi-Step over the other two methods (Figure 6.4 and Appendix Figure E.13). Interestingly, trends with respect to $Sim_P$ are not always consistent with those for $Q_P$, even though the two measure the same aspect of the SCD problem. For example, for the configuration with 100 nodes per snapshot and 16 ground truth segments, even though Multi-Step achieves the highest $Q_P$ score (Figure 6.5(b)), it is the worst-performing method in terms of $Sim_P$ (Appendix Figure E.13). The difference in trends between $Q_P$ and $Sim_P$ is not necessarily suprising, since modularity is known not to always be able to capture well the ground truth communities [221].

6.3.2.3 Overall Solution Quality

For $Sim_B$, SCOUT outperforms the other methods, as it achieves the highest score for 100\% of all synthetic network configurations (Figure 6.6). The other methods are comparable to each other (Figure 6.5(c) and Appendix Figure E.15(c)). Intuitively,
Figure 6.6: Rankings of the methods for synthetic networks with respect to $Sim_B$. Since GHRG could not be run for the larger networks, the results are split into those for the configurations with 50 and 100 nodes per snapshot (top) and those for the configurations with 500 and 1000 nodes per snapshot (bottom). The rankings are computed as follows. For each synthetic network configuration, we compare the four methods’ $Sim_B$ scores (averages over all instances of the given configuration) to identify the first, second, third, and fourth best method; ties are allowed, in which case, two methods would be assigned the same rank. Then, we summarize these results over all considered synthetic network configurations by measuring, for each method ($x$-axis), how many times the given method is ranked as the first, second, third, and fourth best method (expressed as the percentage of all considered configurations; $y$-axis). “N/A” indicates that the given method could not be run (which is the case for GHRG for the larger networks). The figure can intuitively be interpreted as follows: the darker the bar of a given method, the better its performance.
TABLE 6.1

STATISTICAL SIGNIFICANCE OF THE SUPERIORITY OF SCOUT
OVER THE BEST EXISTING METHOD IN TERMS OF $Sim_B$ ON
SYNTHETIC NETWORKS

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Average $Sim_B$</th>
<th>$p$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of nodes</td>
<td># of segments</td>
<td>SCOUT</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.991</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>0.938</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>0.978</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.989</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.966</td>
</tr>
<tr>
<td>500</td>
<td>16</td>
<td>0.931</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.982</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0.960</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.933</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.971</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.937</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>0.902</td>
</tr>
<tr>
<td>500</td>
<td>16</td>
<td>0.841</td>
</tr>
</tbody>
</table>

* “Configuration” columns describe the synthetic network configurations in terms of the number of nodes per snapshot and the number of segments. “Average $Sim_B$” columns contain $Sim_B$ scores of SCOUT and the best of all existing methods (shown in the parentheses), where the scores are averaged over all corresponding synthetic network instances; the highest $Sim_B$ score for a given configuration (i.e., in the given row) is shown in bold. The “$p$-value” column shows the statistical significance of the difference between $Sim_B$ scores of SCOUT and the best existing method. “N/A” means that the scores are identical. $p$-values less than 0.05 are shown in bold, $p$-values less than 0.01 are shown with one star, and $p$-values less than 0.001 are shown with two stars.
the trends with respect to $Sim_B$ seem to follow the trends with respect to segmentation (Section 6.3.2.1) and partition (Section 6.3.2.2) quality aspects of the SCD problem, which is not surprising given that $Sim_B$ captures both of these aspects. When we measure the statistical significance of the improvement of SCOUT over the existing methods, we find that SCOUT statistically significantly improves upon the best existing method in 75%, 65%, and 55% of all cases at $p$-value threshold of 0.05, 0.01, and 0.001, respectively (Table 6.1). Thus, in most of the cases, SCOUT not only improves upon the existing methods but also its improvement is statistically significant. Note that the above percentages could not be perfect, since for 20% of all configurations (namely, the four configurations with the minimum number of ground truth segments), in addition to SCOUT that achieves the perfect $Sim_B$, Multi-Step also (unfairly, per our above discussion) achieves the perfect $Sim_B$ and is thus comparable to SCOUT.
6.3.2.4 Running Time

SCOUT has the lowest running time of all methods, over all synthetic network configurations (Figure 6.7). It is followed by Multi-Step, GraphScope, and GHRG, respectively. Note that GHRG, even when parallelized, cannot be run for the larger networks due to its high computational complexity.

6.3.3 Real-world Networks

We next evaluate the methods on real-world networks. Recall that we consider six real-world networks (Section 6.2.4.2). Since the complete ground truth knowledge (i.e., both change points and segment partitions) is unavailable for any of these networks, we perform evaluation based on $Q_P$ and change point classification (Section 6.3).

We discuss first the segmentation aspect of the SCD problem (change point classification; Section 6.3.3.1) and second the partition aspect of the SCD problem ($Q_P$; Section 6.3.3.2). Third, we compare running times of the methods (Section 6.3.3.3).

6.3.3.1 Segmentation Aspect of the Solution Quality

For change point classification, SCOUT is superior to all of the existing methods, since it achieves the highest accuracy for all considered real-world networks, and it is followed by GHRG, GraphScope, and Multi-Step, respectively (Figure 6.8(a)). This method ranking is consistent with that for synthetic networks (Section 6.3.3.1). Recall that we have formal lists of change points only for four of the six networks (Section 6.2.4.2), and thus the above change point classification is performed only on those four networks. However, we can still intuitively (i.e., informally) discuss segmentation results of the methods for the remaining two networks, High School and Senate, as follows.

Regarding High School network, recall that this network captures proximity of
Figure 6.8: Method comparison for real-world networks with respect to (a) change point classification and (b) $Q_P$. In panel (a), only networks with known ground truth change points are shown, since otherwise change point classification cannot be computed. GHRG could not be run on *AMD Hope* network due to its high computational complexity.
students in a high school (Section 6.2.4.2). Intuitively, for this network, we do not expect large-scale changes in the students’ interaction patterns over time (meaning that we expect very few change points, if any, i.e., very few segments, possibly only one), since students typically interact with other students from the same classes. Consistent with this intuition, SCOUT (as well as GraphScope and Multi-Step) detects only one segment for High School network (Appendix Figure E.17). Moreover, SCOUT (as well as Multi-Step) produces the partition for this single segment that perfectly matches the (static) partition of students according to their classes. Hence, it is encouraging that SCOUT (as well as Multi-Step) captures the above intuition about the expected dynamics and structure of High School network.

Regarding Senate network, for a given method, we identify its 10 top-ranked “change point”-like time points (Section 6.2.4.3). Interestingly, the lists of top ranked time points produced by the different methods have little overlap (Figure 6.9). Specifically, given the four methods and 10 identified points per method, if all methods combined identified only 10 distinct time points, this would mean that the methods produced identical results. On the other hand, if all methods combined identified all \(4 \times 10 = 40\) possible distinct time points, this would mean that the methods produced completely different results with no time point in the overlap of any two methods. In our case, the four methods combined identify 33 out of all 40 possible distinct time points (i.e., 82.5% of them), which means that their results are quite complementary. This is further supported by the fact that there is only one time point that is identified by more than two methods (namely, the 83\(^{rd}\) Congress in 1953, which is among the top 10 ranked time points of SCOUT, GraphScope, and GHRG). We aim to empirically evaluate whether these top ranked points correspond to some important historical events. If so, this would further validate the given method. This evaluation needs to be performed qualitatively (rather than quantitatively, as has been done so far), since it is hard to determine the ranking of all historical events in
terms of their importance and consequently to correlate this ranking with the methods’ ranking of the time points. Because of this, and because the resulting qualitative evaluation is time consuming, while we illustrate the top 10 ranked change points for each method (Figure 6.9), we do not focus here on comparing the different methods. Instead, we focus on discussing SCOUT’s results only, to at least intuitively assess the meaningfulness of its results. SCOUT’s top four time points (1953, 1879, 2003, and 1979, respectively) correspond to Congresses with shifts in the structure of the Senate’s majority between the Democratic and Republican parties. SCOUT’s next three time points correspond to the 86th, 88th, and 67th Congress, respectively. The first two brought major civil rights acts (Civil Rights Act of 1960 and Civil Rights Act of 1964, respectively), and during the third one, “Teapot Dome” Scandal occurred, which is considered one of the most significant investigations in the history of the Senate. SCOUT’s remaining three of the top 10 ranked time points correspond to divided Congresses: the 112th Congress that almost lead to government shutdown, plus the 80th Congress and the 109th Congress, both of which were nicknamed as “do-nothing”. Overall, it is encouraging that SCOUT identifies as likely change points those time points that correspond to important historical events.

1 http://www.senate.gov/history/1921.htm
2 https://en.wikipedia.org/wiki/112th_United_States_Congress
Figure 6.9: The top 10 highest ranked time points by each method for *Senate* network. Time points (expressed in years) are shown on the $x$-axis. Methods are shown on the $y$-axis. Each circle corresponds to one of the top 10 ranked time points, and the darker its color, the higher its rank.
6.3.3.2 Partition Aspect of the Solution Quality

For $Q_P$, with the exception of Hypertext and AMD Hope networks, SCOUT and Multi-Step are comparable, and they outperform both GraphScope and GHRG (Figure 6.8(b)); this is the same trend as for synthetic networks (Section 6.3.2.2). For Hypertext network, SCOUT is outperformed by GHRG and Multi-Step, respectively (Figure 6.8(b)). For AMD Hope network, SCOUT is outperformed by Multi-Step and GraphScope, respectively (Figure 6.8(b)). These results for Hypertext and AMD Hope networks are not necessarily surprising, for the following reason. Different methods can produce solutions with different numbers of segments. In particular, for these two networks, GHRG and GraphScope produce more segments than SCOUT and Multi-Step (Appendix Figure E.17). Recall from Section 6.2.2 that the more segments exist in a solution, the easier it is for this solution to obtain a high partition quality score (i.e., $Q_P$). Hence, a direct comparison of $Q_P$ scores of the solutions with different numbers of segments may not necessarily provide a realistic view of the methods’ performance. As an illustration, consider comparing some two methods: if method 1 has a slightly higher $Q_P$ score than method 2, but it also achieves this score with ten times as many segments as method 2, does it mean that method 1 has a better partition accuracy than method 2? Probably not. Thus, ideally, we would compare $Q_P$ scores of the solutions with equal numbers of segments.

For this reason, since SCOUT is capable of producing a solution with not only an automatically determined but also user-provided number of segments (i.e., since it can solve both SCD and CSCD problems; Section 6.2.2), we compare $Q_P$ score of each existing method and $Q_P$ score of SCOUT when solving the $Q_P$-based CSCD problem and producing a solution with the same number of segments as the solution of the given existing method. In this way, we avoid the bias arising from the fact that the two compared methods might have different numbers of segments. According to this evaluation, SCOUT outperforms all methods (Figure 6.10 and Appendix...
Figure 6.10: Representative $Q_P$ scores of 1) SCOUT’s solutions for different numbers of segments $l$ and 2) the solutions of the existing methods, for (a) High School network and (b) Senate network. For SCOUT, the line shows its $Q_P$ score when solving the $Q_P$-based CSCD problem while varying the number of segments. For each of the existing methods, the mark shows $Q_P$ score of the given method’s solution, with the position of the mark along the $x$-axis corresponding to the number of segments $l$ in the solution. Equivalent results for the remaining real-world networks are shown in Appendix Figure E.18.
Figure 6.11: Running times of the methods for real-world networks (logarithmic scale). The striped bars mean that the method could not finish within the allowed time.

Figure E.18).

The shape of the $Q_P$-curve as a function of the number of segments $l$ could provide insights into the dynamics of the network in question. Even though the $x$-axis of the curve does not correspond to time, and thus it cannot tell us when changes in community organization (if any) occur, the fact that the $x$-axis corresponds to $l$ can intuitively tell us something about the number of such changes and their scale. Namely, on the one hand, if $Q_P$ increases slowly (or does not increase at all) as $l$ increases, this could mean that the community organization of the network does not change a lot with increase in the number of segments, and thus, the increase in the number of segments is unnecessary. For example, this is the case for High school network (Figure 6.10(a)), which agrees with our discussion in Section 6.3.3.1. On the other hand, if $Q_P$ increases drastically as $l$ increases, this could mean that the community organization of the network indeed changes a lot with increase in the
number of segments, and thus, the increase in the number of segments is justified. For example, this is the case for Senate network (Figure 6.10(b)), which agrees with our discussion in Section 6.3.3.1.

6.3.3.3 Running Time

Just as for synthetic networks, SCOUT has the lowest running time of all methods, over all real-world networks (Figure 6.11). Again, GHRG is the slowest among all considered methods, which means that it cannot be run for the larger networks due to its high computational complexity.

6.4 Conclusions

We study the problem of community detection in dynamic networks. To capture the intuition of a compromise between the two extremes of snapshot clustering and consensus clustering, we combine community detection with the problem of segment detection to formulate a new problem of SCD. To address the drawbacks of the existing methods that can be employed to solve the SCD problem, we introduce SCOUT. To comprehensively evaluate SCOUT against the existing methods, we introduce a synthetic network generator that produces a dynamic network with the known ground truth segments and their community organization, where by varying the model parameters, different synthetic dynamic network configurations can be obtained. To quantify the performance of a given method, we introduce new measures of SCD quality. We perform our experiments on a variety of synthetic as well as real-world networks. We demonstrate that SCOUT outperforms the existing methods with respect to both segmentation aspect and partition aspect of the SCD problem. At the same time, SCOUT is more computationally efficient than the existing methods. Ultimately, we show that the SCD problem and SCOUT in particular is a useful framework for studying community organization of dynamic networks, as it can
identify both when communities evolve by identifying change points and how communities look like at each stage of their evolution by identifying segment partitions. The solution of the SCD problem provides a concise yet informative description of the dynamic network from the perspective of its community organization.

Our work has several potential future directions. From the methodological perspective, SCOUT could be extended to different problem settings, such as dealing with weighted networks or overlapping communities. From the application perspective, an important problem in dynamic network analysis is to choose a meaningful time scale for defining network snapshots. Usually, the time scale is chosen so that each snapshot is assumed to have the same duration (e.g., one week), and the duration is determined empirically to fit the context of the given application. Instead, the output of the SCD problem could provide a systematic way for defining snapshots. Namely, the smallest meaningful traditional empirical equal-length snapshots would be used to define the initial dynamic network. Then, this network would be given as input to SCOUT to group the small snapshots with consistent community organization into larger segments. Finally, the time interval of each segment would correspond to a new, more meaningful snapshot, and collection of all such new snapshots would form a new, more meaningful dynamic network. In this way, each snapshot of the new network would capture the period during which community organization is consistent. Moreover, the duration of different snapshots could be different. These newly constructed snapshots (i.e., the new dynamic network) could then be used as input to various methods for dynamic network analysis, which could improve the quality of results compared to using the same methods on the traditionally determined empirical same-length snapshots (i.e., on the initial network that was given as input to SCOUT).
6.5 Computational Novelty of the Work

We propose novel algorithmic directions for studying community organization of dynamic networks. In particular, we formulate the SCD problem whose goal is to simultaneously group snapshots of the network into segments according to their community organization and group nodes of the snapshots into clusters based on the segments these snapshots belong to. To address the limitations of the existing methods, we introduce SCOUT. For experimental evaluation, we introduce synthetic network generator, SCG, that can generate dynamic networks with embedded ground truth. We also introduce new measures of SCD quality for quantifying the performance of the methods. In a thorough evaluation on both synthetic and real-world networks, we demonstrate that SCOUT outperforms the existing methods in terms of both accuracy and running time. We plan to make our implementations publicly available when the SCOUT manuscript is accepted for publication.

6.6 My Contributions to the Work

I helped design the study, designed and implemented the methods, carried out all computational analyses, helped analyze the results, and helped write the manuscript.
In this dissertation, we introduce novel computational strategies for exploring structure and dynamics of complex networks. We organize our contributions according to two different perspectives: answering novel research questions via established network approaches (Part I) and developing novel network approaches for established research questions (Part II).

In the first context, we start by using network analysis to answer a novel question in a novel domain in which network research has not been used to date – interpreting affective physiological data (Chapter 2). Studying this is important because understanding how human physiological responses to stimuli vary across individuals is critical for the field of affective computing. As a result, we introduce a framework for systematic network analysis of human physiological responses and show that it outperforms an alternative non-network-based approach. Our work makes initial steps toward the ultimate goal of understanding how different emotions are manifested in physiological signals both within and across individuals, which is critical for the field of affective computing.

We also study the interplay between individuals’ social interactions and traits from a new viewpoint of dynamic social network data (Chapter 3). Studying this is important to answer to what extent does one shape the social network(s) one participates in, or to what extent do the social networks shape the individual. We uncover a number of existing network-trait relationships, which validates our results. At the same time, we obtain additional insights compared to static network analysis.
alone or compared to the existing studies. Our framework can be extended to predict traits of uncharacterized nodes based on their network similarities to characterized nodes. This is important, since in many real-world applications it is easier to measure interaction data than nodes’ traits.

In the second context, we start by using the well-established concept of graphlets to develop a novel method for a popular computational problem – that of link prediction (Chapter 4). Our methodology not only accounts for extended topology but also allows for a fair evaluation of important link prediction parameters. Importantly, when we use the link prediction methods to de-noise real-world cellular networks, we find that the de-noised networks improve biological correctness of the original networks, which is the ultimate goal of link prediction in computational biology. Thus, computational de-noising of current cellular networks could lead to better topology-based predictions and consequently save resources needed for experimental validation of the predictions.

Moreover, we take the established notion of static graphlets to the next level to develop new graphlet-based theory that allows for dynamic network analysis (Chapter 5). Our results demonstrate that the resulting notion of dynamic graphlets is superior compared to static and even static-temporal graphlet-based approaches when distinguishing between different network types or functionally different nodes. Hence, dynamic graphlets could lead to more effective analyses of evolving networks. We apply our dynamic graphlet approach to biological networks to deepen our currently limited knowledge about human aging. Importantly, just as static graphlets have been used for a variety of computational tasks, such as network clustering, network comparison and alignment, or link prediction, as well as applied tasks, such as studying disease and host-pathogen interactions, as more dynamic network data is becoming available, dynamic graphlets are promising in all of these contexts as well.

Finally, we combine the problems of community detection and segment detection
to study community organization of dynamic networks (Chapter 6). As a result, we present SCOUT, a new framework that simultaneously groups consecutive snapshots of a dynamic network with similar community organization into segments and finds community organization for each segment that fits well all of its snapshots. We demonstrate that SCOUT outperforms the existing methods in terms of accuracy as well as computational complexity. There are several directions for future work regarding SCOUT. Given its flexibility, SCOUT could be extended to handle various problem formulations, such as weighted or directed networks and overlapping or hierarchical communities. Of particular interest is to allow for handling not only snapshot-based dynamic network representations, but also representations directly in the form of temporal edges. In fact, choosing a meaningful time scale for defining network snapshots is an important problem in dynamic network analysis. SCOUT could provide a systematic way for approaching this problem, starting with either the smallest meaningful snapshots or directly with temporal edges, and using the produced segments as a basis for defining more meaningful snapshots. Another direction is to capture relationships between adjacent segments, each of which is currently studied in isolation. Even though different segments are assumed to have different community organizations, it does not necessarily mean that these segments are independent of each other. Thus, perhaps even by leveraging the notion of dynamic graphlets, it could be of interest to model and account for inter-snapshot relationships.

Having summarized our contributions and outlined their potential extensions, we conclude by discussing how individual parts of the dissertation fit together and reflecting upon a longer-term vision for the future. The ultimate motivation of our research is understanding the link between the structure/dynamics of real-world networks and their function. Towards this goal, in Part I, we answer applied research questions via established network approaches, focusing on network analyses of human physiological responses (Chapter 2) and social interactions (Chapter 3). In the pro-
cess, we thoroughly evaluate the existing approaches in the context of different types of network data (static and dynamic, respectively) and structural levels (global and local, respectively). Importantly, while we show that network-based analyses provide useful insights into the applied questions at hand and the corresponding domains, we also deepen our understanding of the current network approaches and their drawbacks. In Part II, we aim to leverage this understanding to develop novel network approaches that address limitations of the existing work and that are applicable to various domains. In particular, we use graphlets to develop new methods for link prediction in static networks (Chapter 4) and characterizing local and global structure in dynamic networks (Chapter 5). Moreover, we introduce a new approach for studying intermediate structure (via community organization) in dynamic networks (Chapter 6). Hence, we explore networks from various structural (from global to intermediate to local) and temporal (from static to dynamic) viewpoints.

The two perspectives of network science presented in the dissertation – using existing network approaches in novel domains (Part I) and developing novel network approaches (Part II) – are not independent, but instead complement and motivate each other. As the technologies for collecting interconnected data in various domains advance, it becomes increasingly important to be able to use the collected network data for answering new research questions in these domains. Some prominent examples include understanding how the brain works, uncovering the link between aging and disease, studying adoption of behaviors among individuals, and characterizing information pathways in social communications. Importantly, the connection between networks and systems they model goes both ways: network structure does not merely reflect the function of the system, but can also be used to influence it. For example, when studying the spread of behaviors in a social network, one could go from simply observing to actively modifying the network with the goal of promoting the adoption of healthy habits to make a positive impact.
The increasing availability of the network data in various domains is also accompanied by their growth in size and complexity, with one important aspect of the latter being the time dimension. A key to comprehending such data is to capture how interactions unfold over time, going beyond simple static graph models, and to do so in a scalable way. While studying dynamic networks was one of the key focuses of our work, there are many open research questions and unexplored directions in this area. For example, what is the meaning of a time scale of a dynamic network, and is there one “true” scale of the network or does its appropriate definition depend on a specific application (e.g., spread of contagious diseases or adoption of opinions), structural level (e.g., global or local), or even part of the network (e.g., for some nodes, interactions remain unchanged, while for others, they change over time)? What is a community in the dynamic setting, and how should it integrate topological and temporal aspects of network structure? How to visualize and summarize a dynamic network in an intuitive and informative way? Moreover, in addition to evolving over time, real-world networks may contain multiple types of nodes and edges, as well as various attributes for them, and accounting for this rich information could lead to further insights.

In summary, exploring the structure and dynamics of real-world networks presents numerous exciting research challenges as well as opportunities, leading to high-impact applications and advancing our understanding of the interconnected world.
APPENDIX A

FOR CHAPTER 2

A.1 IAPS Identifiers

Nine classes of the AV “ground truth” partition, illustrated in Figure 2.2(a), contain images with the following IAPS identifiers: 1) negative valence, low arousal: 2039, 2104, 2440, 2722, 5130, 7040, 9260, 9291, 9331, 9390; 2) negative valence, medium arousal: 1111, 1270, 2095, 2456, 3301, 6241, 9005, 9320, 9440, 9831; 3) negative valence, high arousal: 1050, 1205, 2730, 3000, 3212, 3213, 6520, 7380, 9075; 4) neutral valence, low arousal: 2190, 2397, 2499, 2720, 5875, 6150, 7041, 7255, 7287, 9700; 5) neutral valence, medium arousal: 1122, 1350, 2034, 2752, 2770, 2780, 7079, 9469, 9582, 9594; 6) neutral valence, high arousal: 1113, 1302, 1726, 1931, 3211, 3302, 4008, 4604, 7640, 9230; 7) positive valence, low arousal: 1605, 1610, 1620, 2000, 2370, 2501, 5200, 5760, 5811, 7325; 8) positive valence, medium arousal: 1463, 1540, 1630, 1920, 2071, 2092, 2345, 2352, 8205, 8350; 9) positive valence, high arousal: 2208, 2347, 4180, 4693, 5480, 8179, 8206, 8496, 8500, 8502.
APPENDIX B

FOR CHAPTER 3

B.1 Appendix Figures

Figure B.1: Statistics of a network of the given type (S, P, or B) at the given timeslot in terms of: (a) the percentage of all the nodes that are in the network’s largest connected component (LCC) and (b) network density, for the friends and family dataset.
Figure B.2: Statistically significantly correlated trait pairs according to at least one of AID and WRST for the *friends and family* dataset (p-value threshold of 0.01). We consider as “correlated ground truth pairs” (i.e., as positive controls for validation purposes) those trait pairs that are statistically significantly correlated with respect to both AID and WRST (solid lines in the graph). We consider as “uncorrelated ground truth pairs” (i.e., as negative controls for validation purposes) those trait pairs that are not statistically significantly correlated with respect to either AID or WRST (non-edges in the graph). There are 10 trait pairs in total, of which two are correlated positive control pairs, seven are uncorrelated negative control pairs, and the remaining one pair is neither positive nor negative control pairs, as it is correlated with exactly one of AID or WRST.

Figure B.3: Interplays between a network type (see the given matrix column) and a trait (see the given matrix row) for the *NetSense* dataset and the *equal size* trait-based partitioning strategy detected via at least one centrality by AID-ED based on: (a) dynamic network analysis (black) and (b) dynamic network analysis only (black), static network analysis only (striped), both dynamic and static network analysis (grey), or none of dynamic and static network analysis (white).
Figure B.4: Interplays between a network type (see the given matrix column) and a trait (see the given matrix row) for the friends and family dataset and the k-medoids trait-based partitioning strategy detected via at least one centrality by AID-ED based on: (a) dynamic network analysis (black) and (b) dynamic network analysis only (black), static network analysis only (striped), both dynamic and static network analysis (grey), or none of dynamic and static network analysis (white).

Figure B.5: Interplays between a network type (see the given matrix column) and a trait (see the given matrix row) for the friends and family dataset and the equal size trait-based partitioning strategy detected via at least one centrality by AID-ED based on: (a) dynamic network analysis (black) and (b) dynamic network analysis only (black), static network analysis only (striped), both dynamic and static network analysis (grey), or none of dynamic and static network analysis (white).
### TABLE B.1

VALIDATION RATES OF OUR RESULTS ON THE *NetSense* DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITY, FOR EACH COMBINATION OF THE TWO STATISTICAL METHODS (AID AND WRST), CLUSTERING DISTANCE METRICS (ED AND PC), AND NETWORK ANALYSIS TYPES (DYNAMIC AND STATIC)

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Metric</th>
<th>Correlated</th>
<th>Uncorrelated</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic AID ED</td>
<td>43% (3/7)</td>
<td>5% (2/38)</td>
<td>714%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic AID PC</td>
<td>14% (1/7)</td>
<td>18% (7/38)</td>
<td>-22%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic WRST ED</td>
<td>86% (6/7)</td>
<td>58% (22/38)</td>
<td>48%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic WRST PC</td>
<td>71% (5/7)</td>
<td>61% (23/38)</td>
<td>18%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static AID ED</td>
<td>29% (2/7)</td>
<td>16% (6/38)</td>
<td>81%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static WRST ED</td>
<td>86% (6/7)</td>
<td>76% (29/38)</td>
<td>12%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The denominators correspond to the total number of correlated (for positive controls) and uncorrelated (for negative controls) trait pairs, and the nominators correspond to the number of respective trait pairs captured by our approach. The improvement is measured as the difference between positive control validation rate and negative control validation rate, divided by negative control validation rate. Note that all reported percentages are rounded.
### Table B.2

Validation rates of our results on the NetSense dataset in terms of expected trait-trait (dis)similarity, when taking into account only the Big Five personality traits only, for each combination of the two statistical methods (AID and WRST), clustering distance metrics (ED and PC), and network analysis types (dynamic and static).

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Metric</th>
<th>Correlated</th>
<th>Uncorrelated</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic</td>
<td>AID</td>
<td>ED</td>
<td>100% (1/1)</td>
<td>33% (2/6)</td>
<td>200%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>AID</td>
<td>PC</td>
<td>0% (0/1)</td>
<td>17% (1/6)</td>
<td>-100%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>WRST</td>
<td>ED</td>
<td>100% (1/1)</td>
<td>67% (4/6)</td>
<td>50%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>WRST</td>
<td>PC</td>
<td>100% (1/1)</td>
<td>100% (6/6)</td>
<td>0%</td>
</tr>
<tr>
<td>Static</td>
<td>AID</td>
<td>ED</td>
<td>0% (0/1)</td>
<td>0% (0/6)</td>
<td>-</td>
</tr>
<tr>
<td>Static</td>
<td>WRST</td>
<td>ED</td>
<td>0% (0/1)</td>
<td>83% (5/6)</td>
<td>-100%</td>
</tr>
</tbody>
</table>

* The denominators correspond to the total number of correlated (for positive controls) and uncorrelated (for negative controls) trait pairs, and the nominators correspond to the number of respective trait pairs captured by our approach. The improvement is measured as the difference between positive control validation rate and negative control validation rate, divided by negative control validation rate. Note that all reported percentages are rounded.
TABLE B.3

VALIDATION RATES OF OUR RESULTS ON THE friends and family DATASET IN TERMS OF EXPECTED TRAIT-TRAIT (DIS)SIMILARITY, WHEN TAKING INTO ACCOUNT ONLY THE BIG FIVE PERSONALITY TRAITS ONLY, FOR EACH COMBINATION OF THE TWO STATISTICAL METHODS (AID AND WRST), CLUSTERING DISTANCE METRICS (ED AND PC), AND NETWORK ANALYSIS TYPES (DYNAMIC AND STATIC)

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Metric</th>
<th>Correlated</th>
<th>Uncorrelated</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic</td>
<td>AID</td>
<td>ED</td>
<td>50% (1/2)</td>
<td>57% (4/7)</td>
<td>-13%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>AID</td>
<td>PC</td>
<td>50% (1/2)</td>
<td>43% (3/7)</td>
<td>17%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>WRST</td>
<td>ED</td>
<td>100% (2/2)</td>
<td>43% (3/7)</td>
<td>133%</td>
</tr>
<tr>
<td>Dynamic</td>
<td>WRST</td>
<td>PC</td>
<td>50% (1/2)</td>
<td>14% (1/7)</td>
<td>250%</td>
</tr>
<tr>
<td>Static</td>
<td>AID</td>
<td>ED</td>
<td>0% (0/2)</td>
<td>0% (0/7)</td>
<td>-</td>
</tr>
<tr>
<td>Static</td>
<td>WRST</td>
<td>ED</td>
<td>100% (2/2)</td>
<td>57% (4/7)</td>
<td>75%</td>
</tr>
</tbody>
</table>

* The denominators correspond to the total number of correlated (for positive controls) and uncorrelated (for negative controls) trait pairs, and the nominators correspond to the number of respective trait pairs captured by our approach. The improvement is measured as the difference between positive control validation rate and negative control validation rate, divided by negative control validation rate. Note that all reported percentages are rounded.
C.1 Density-weighted Node-pair-GDV-centrality

As discussed in Section 4.2.3.3, we aim to favor denser shared graphlets over sparser shared graphlets, where we define the density of a graphlet as the percentage of edges present in the graphlet out of all possible edges, ignoring the node pair in question. Thus, we design density-weighted node-pair-GDV-centrality, as follows. If $c_i$ is the $i^{th}$ element of node-pair-GDV of the nodes $v$ and $u$, then we define density-weighted node-pair-GDV-centrality of $v$ and $u$ as:

$$ \sum_{i=0}^{49} w_i \times (\frac{3}{2}d_i - \frac{1}{2})^{10} \times \log(c_i + 1), $$

where $d_i$ is the density of the graphlet corresponding to node pair orbit $i$ (and $w_i$ is the weight that takes into account orbit dependencies, just as in the original definition).

This is just one possible heuristic for favoring denser graphlets over sparser ones. In the formula, the density is multiplied by $\frac{3}{2}$ and then subtracted by $\frac{1}{2}$ to scale it to $[0,1]$ interval. The scaled density is then raised to the power of 10 to favor higher density graphlets (especially cliques) over lower density graphlets exponentially rather than linearly.

C.2 Evaluation Framework

As discussed in Section 4.2.4, we evaluate each of the existing and new LP methods on each of the PPI networks by computing the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) predicted by the method. TP are predicted edges that are present in the original PPI network, TN are
missed edges (i.e., predicted non-edges) that are absent from the original PPI network, FP are predicted edges that are absent from the original PPI network, and FN are missed edges that are present in the original PPI network. Clearly, an ideal method would maximize TP and TN and minimize FP and FN over the entire range of \( k \). We summarize the four statistics in two ways, by computing: 1) precision and recall, together with the corresponding F-score and 2) sensitivity and specificity, together with the corresponding receiver-operator curves (ROCs). Precision = \( \frac{TP}{TP+FP} \). Recall = \( \frac{TP}{TP+FN} \). As precision increases, recall decreases, and vice versa. To balance between the two measures, we combine them into popular F-score measure as: 

\[
F-score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.
\]

We plot precision-recall and F-score curves over the entire range of \( k \). Sensitivity = \( \frac{TP}{TP+FN} \). Specificity = \( \frac{TN}{FP+TN} \). For simplicity of comparing results across different methods, we summarize the performance of the methods over the entire range of \( k \) with respect to sensitivity and specificity by calculating the areas under the ROCs (AUROCs).

After we evaluate the LP methods, we apply them to a PPI network to de-noise it, and we evaluate the biological quality of the de-noised network with respect to the “enrichment” of predicted edges in Gene Ontology (GO) terms. That is, we compute the enrichment as the percentage of predicted edges, out of all edges in which both proteins have at least one GO term, in which the two end nodes share a GO term. We compute the statistical significance of the given enrichment by calculating the probability (i.e., \( p \)-value) of observing the same or higher enrichment by chance using the hypergeometric model. Let us denote by \( N \) the set of pairs of proteins in the original PPI network where both proteins have at least one biological process GO term, by \( S \) the subset of protein pairs in \( N \) that have a GO term in common, by \( e \) the subset of protein pairs in \( N \) that are predicted as edges, and by \( k \) the subset of protein pairs in \( e \) that have a GO term in common. Then, the enrichment is \( \frac{|k|}{|e|} \) and its \( p \)-value is: \( p-value = 1 - \sum_{i=0}^{|k|-1} \frac{\binom{|S|}{i} \binom{|N|-|S|}{|k|-i}}{\binom{|e|}{i}} \). We use \( p \)-value threshold of 0.05.
Also, we validate predicted edges absent from the original network by searching for them in an independent PPI data source. We measure the statistical significance of validating the given number of predictions by using the above formula, except that now $N$ is the set of all pairs of proteins in the PPI network in which both proteins are present in the BioGRID data, $S$ is the subset of protein pairs in $N$ that interact in BioGRID, $e$ is the subset of protein pairs in $N$ that are new predicted edges, and $k$ is the subset of protein pairs in $e$ that interact in BioGRID. We use $p$-value threshold of 0.05.
Figure C.1: Link prediction accuracy of the unweighted graphlet-based methods in terms of AUROCs at different noise levels as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.2: Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUROCs at different noise levels as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.3: Link prediction accuracy of the graphlet-based methods in terms of AUROCs at different noise levels comparing unweighted and weighted versions of the methods. The first column corresponds to using 3-5-node graphlets and the second column corresponds to using 3-4-node graphlets. The results for using 3-node-graphlets are not included, as for this graphlet size, the unweighted and weighted versions of the methods are equivalent. Also, unlike in the following figures, the results for $\alpha$ of 0 are not included in this figure, as at this $\alpha$, the unweighted and weighted versions of the methods are equivalent. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.4: Link prediction accuracy of the weighted graphlet-based methods in terms of AUROCs at different noise levels comparing different graphlet sizes in each of the three networks (AP/MS, HC, and Y2H). The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.5: Link prediction accuracy of our new methods and existing methods in terms of AUROCs at different noise levels comparing the different methods in each of the three networks (AP/MS, HC, and Y2H). The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.6: Receiver operator curves for the AP/MS network for each noise level. The vertical error bar at a given value of $k$ and for a given method corresponds to the standard deviation of true positive rate over five random runs of the method. The horizontal error bar at a given value of $k$ and for a given method corresponds to the standard deviation of false negative rate over five random runs of the method.
Figure C.7: Receiver operator curves for the HC network for each noise level. The vertical error bar at a given value of \( k \) and for a given method corresponds to the standard deviation of true positive rate over five random runs of the method. The horizontal error bar at a given value of \( k \) and for a given method corresponds to the standard deviation of false negative rate over five random runs of the method.
Figure C.8: Receiver operator curves for the Y2H network for each noise level. The vertical error bar at a given value of $k$ and for a given method corresponds to the standard deviation of true positive rate over five random runs of the method. The horizontal error bar at a given value of $k$ and for a given method corresponds to the standard deviation of false negative rate over five random runs of the method.
Figure C.9: Link prediction accuracy of the unweighted graphlet-based methods in terms of AUPRs at different noise levels as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.10: Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUPRs at different noise levels as \( \alpha \) is varied from 0 to 1. Recall that \( \alpha = 0 \) corresponds to using node-GDV-similarity alone and \( \alpha = 1 \) corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.11: Link prediction accuracy of the graphlet-based methods in terms of AUPRs at different noise levels comparing unweighted and weighted versions of the methods. The first column corresponds to using 3-5-node graphlets and the second column corresponds to using 3-4-node graphlets. The results for using 3-node-graphlets are not included, as for this graphlet size, the unweighted and weighted versions of the methods are equivalent. Also, unlike in the following figures, the results for $\alpha$ of 0 are not included in this figure, as at this $\alpha$, the unweighted and weighted versions of the methods are equivalent. The first row corresponds to the AP/MS network, the second row corresponds to the HC network, and the third row corresponds to the Y2H network. The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.12: Link prediction accuracy of the weighted graphlet-based methods in terms of AUPRs at different noise levels comparing different graphlet sizes in each of the three networks (AP/MS, HC, and Y2H). The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.13: Link prediction accuracy of our new methods and existing methods in terms of AUPRs at different noise levels comparing the different methods in each of the three networks (AP/MS, HC, and Y2H). The error bar for a given method corresponds to the standard deviation over five random runs of the method. In some cases, standard deviations are so small that they are not visible.
Figure C.14: Precision-recall curves for the AP/MS network for each noise level. The vertical error bar at a given value of $k$ and for a given method corresponds to the standard deviation of precision over five random runs of the method. The horizontal error bar at a given value of $k$ and for a given method corresponds to the standard deviation of recall over five random runs of the method.
Figure C.15: Precision-recall curves for the HC network for each noise level. The vertical error bar at a given value of $k$ and for a given method corresponds to the standard deviation of precision over five random runs of the method. The horizontal error bar at a given value of $k$ and for a given method corresponds to the standard deviation of recall over five random runs of the method.
Figure C.16: Precision-recall curves for the Y2H network for each noise level. The vertical error bar at a given value of $k$ and for a given method corresponds to the standard deviation of precision over five random runs of the method. The horizontal error bar at a given value of $k$ and for a given method corresponds to the standard deviation of recall over five random runs of the method.
Figure C.17: F-score curves for the AP/MS network for each noise level.
Figure C.18: F-score curves for the HC network for each noise level.
Figure C.19: F-score curves for the Y2H network for each noise level.
Figure C.20: Link prediction accuracy of the unweighted graphlet-based methods in terms of AUROCs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only.

Figure C.21: Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUROCs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only.
Figure C.22: Link prediction accuracy of the graphlet-based methods in terms of AUROCs for the HC network against the low confidence data comparing unweighted and weighted versions of the methods. The first column corresponds to using 3-5 node graphlets and the second column corresponds to using 3-4 node graphlets. The results for using 3-node graphlets are not included, as for this graphlet size, the unweighted and weighted versions of the methods are equivalent. We identify from Appendix Figures C.20 and C.21 the values of $\alpha$ which result in the highest AUROCs, and we include these $\alpha$s (namely 0.2 and 0.4) into the figure. Also, we include $\alpha$ of 0.8, as this $\alpha$ was the best among all $\alpha$s when evaluating the methods on noisy networks (Appendix Figures C.1 and C.2). In addition, we include $\alpha$ of 1, as this $\alpha$ corresponds to using only node-pair-GDV-centrality and no node-GDV-similarity in the total link prediction score.
Figure C.23: Link prediction accuracy of the graphlet-based methods in terms of AUROCs for the HC network against the low confidence data comparing different graphlet sizes. We include into this figure the same $\alpha$s as in Appendix Figure C.22.

Figure C.24: Receiver operator curves for the HC network against the low confidence data comparing the different methods.
Figure C.25: Link prediction accuracy of the unweighted graphlet-based methods in terms of AUPRs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only.

Figure C.26: Link prediction accuracy of the density-weighted graphlet-based methods in terms of AUPRs for the HC network against the low confidence data as $\alpha$ is varied from 0 to 1. Recall that $\alpha = 0$ corresponds to using node-GDV-similarity alone and $\alpha = 1$ corresponds to using node-pair-GDV-centrality alone. The first column corresponds to using 3-5-node graphlets, the second column corresponds to using 3-4-node graphlets, and the third column corresponds to using 3-node graphlets only.
Figure C.27: Link prediction accuracy of the graphlet-based methods in terms of AUPRs for the HC network against the low confidence data comparing unweighted and weighted versions of the methods. The first column corresponds to using 3-5-node graphlets and the second column corresponds to using 3-4-node graphlets. The results for using 3-node-graphlets are not included, as for this graphlet size, the unweighted and weighted versions of the methods are equivalent. We identify from Appendix Figures C.20 and C.21 the values of $\alpha$ which result in the highest AUPRs, and we include these $\alpha$s (namely 0.2 and 0.4) into the figure. Also, we include $\alpha$ of 0.8, as this $\alpha$ was the best among all $\alpha$s when evaluating the methods on noisy networks (Appendix Figures C.1 and C.2). In addition, we include $\alpha$ of 1, as this $\alpha$ corresponds to using only node-pair-GDV-centrality and no node-GDV-similarity in the total link prediction score.
Figure C.28: Link prediction accuracy of the graphlet-based methods in terms of AUPRs for the HC network against the low confidence data comparing different graphlet sizes. We include into this figure the same $\alpha$s as in Appendix Figure C.27.

Figure C.29: Precision-recall curves for the HC network against the low confidence data comparing the different methods.
Figure C.30: Pairwise intersections between each pair of de-noised networks as well as between each de-noised network and the original network, for each of the three networks (AP/MS, HC, and Y2H).
Figure C.31: Percentage of new predicted edges from AP/MS that are validated in BioGRID.
## C.4 Appendix Tables

### TABLE C.1

A SAMPLE OF THE LIST OF ALL NODE PAIRS PREDICTED AS EDGES BY ANY METHOD

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th># of methods</th>
<th>New predicted edge</th>
<th>DP</th>
<th>SN</th>
<th>JC</th>
<th>AA</th>
<th>Katz</th>
<th>LPI</th>
<th>RAI</th>
<th>RWS</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>YLR441C</td>
<td>YKR081C</td>
<td>4</td>
<td>1</td>
<td>5978</td>
<td>8303</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6942</td>
</tr>
<tr>
<td>YPL131W</td>
<td>YAL035W</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>8303</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7983</td>
</tr>
<tr>
<td>YJR002W</td>
<td>YDR496C</td>
<td>1</td>
<td>1</td>
<td>6184</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YGL127C</td>
<td>YBL093C</td>
<td>8</td>
<td>0</td>
<td>-</td>
<td>5112</td>
<td>1284</td>
<td>4706</td>
<td>4493</td>
<td>6060</td>
<td>3530</td>
<td>426</td>
<td>4745</td>
</tr>
<tr>
<td>YLR399C</td>
<td>YDR485C</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>4980</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2718</td>
</tr>
</tbody>
</table>

*Note that this is just a sample (top five rows for the AP/MS network) of the entire table that is available online at [http://www.nd.edu/~cone/LP/ST1.xlsx](http://www.nd.edu/~cone/LP/ST1.xlsx). We provide a sample only since the entire table is too large to be directly incorporated into the Appendix. In the table, Columns 1 and 2 correspond to the node names. Column 3 corresponds to the number of methods that predict the given node pair as an edge. Column 4 contains a “1” if the node pair is a “new predicted edge” (predicted edge not present in the original network) and a “0” otherwise. Columns 5-13 list the rankings of the given node pair by the different methods (the same methods as in Figure 4.3(a)); if the given node pair is not predicted as an edge by a given method, a ‘-’ appears. The three tabs/sheets in the online file containing the entire table correspond to the three analyzed networks: AP/MS, HC, and Y2H.*
D.1 Computing the Number of Dynamic Graphlet Types of a Given Size

Here, we expand the discussion from Section 5.2.1 on how to compute \( D(n,k) \), the number of dynamic graphlet types with \( n \) nodes and \( k \) events.

Since at least \( n - 1 \) edges are needed to connect \( n \) nodes, \( D(n,k) = 0 \) for \( k < n - 1 \). Moreover, since our events are undirected, \( D(2,k) = 1 \), for any \( k \). To compute \( D(n,k) \) when \( n \geq 3 \) and \( k \geq n - 1 \), notice that each dynamic graphlet with \( k \) events can be formed from a dynamic graphlet with \( k - 1 \) events and either \( n - 1 \) or \( n \) nodes, by adding a new event between some two existing nodes or between an existing node and a new node, respectively.

In the first case, we take a dynamic graphlet with \( n \) nodes and \( k - 1 \) events and add a new event between its existing nodes, in order to obtain a dynamic graphlet with \( n \) nodes and \( k \) events (e.g., construct \( D_6 \) from \( D_2 \)). Due to the \( \Delta t \)-connectivity constraint, this new event has to involve at least one of the two nodes participating in event \((k - 1)\). We can add the new event in \( 2n - 3 \) different ways: between one of these nodes and the “remaining” \( n - 2 \) nodes (which is \( 2(n - 2) = 2n - 4 \) ways) or just duplicate event \((k - 1)\).

In the second case, we take a dynamic graphlet with \( n - 1 \) nodes and \( k - 1 \) events and add an event from one of its nodes to the new \((n^{th})\) node, in order to obtain a dynamic graphlet with \( n \) nodes and \( k \) events (e.g., construct \( D_4 \) from \( D_1 \)). For \( n - 1 \geq 3 \), there are two ways to do this, since there are two potential candidates.
for this new event (the two endpoints of event \((k - 1)\)). Note that since events are undirected, for \(n - 1 = 2\), the two nodes are indistinguishable from our point of view, and so we have only one way to construct a new dynamic graphlet with 3 nodes and \(k\) events.

In summary, we can get \(2n - 3\) new dynamic graphlets with \(n\) nodes and \(k\) events from each dynamic graphlet with \(n\) nodes and \(k - 1\) events. Moreover, we can get two new dynamic graphlets with \(n\) nodes and \(k\) events from each dynamic graphlet with \(n - 1\) nodes and \(k - 1\) events. The only exception is for \(n = 3\), since we can get only one new dynamic graphlet with three nodes from a dynamic graphlet with two nodes, as these two nodes are indistinguishable. Importantly, since each dynamic graphlet with \(k\) events has a unique \((k - 1)\)-“prefix” from which it was extended, all of these new dynamic graphlets with \(n\) nodes will be different. Thus, we get the following recursive formulas for \(D(n, k)\): \(D(3, k) = 3D(3, k - 1) + D(2, k - 1), n = 3;\) and \(D(n, k) = (2n - 3)D(n, k - 1) + 2D(n - 1, k - 1), n > 3.\) By expanding the formulas for few smallest values of \(n\) and \(k\), we get the following closed-form solution (Appendix Table D.2):

\[
D(n, k) = \sum_{i=0}^{n-2} \frac{(-1)^{n+i} \binom{n-2}{i} (2i + 1)^{k-1}}{2(n - 2)!}, n \geq 3.
\]

D.2 Constrained Counting of Dynamic Graphlets in a Network

Here, we expand the discussion from Section 5.2.2 on the constrained dynamic graphlet counting procedure.

A network having dense neighborhoods with many events between the same nodes will have large counts of dynamic graphlet types. This is because for a given dynamic graphlet instance, there will be many \(\Delta t\)-adjacent candidate events which can be used to “grow” this graphlet. For each of these possible graphlet extensions, we will again have many possibilities for further extension, and so on. For example, consider a
snapshot-based network representation where each snapshot is the same dense graph.
A large number of different dynamic graphlet instances will be detected, yet many of
them will just be artifacts of the consecutive snapshots “sharing” the dense network
structure. As an (rather extreme) example, consider all snapshots being the same
fully connected graph. In this case, for a dynamic graphlet instances ending with
some event, we will have multiple events that can be used to grow this graphlet,
resulting in large counts for all possible dynamic graphlet types. Clearly, considering
all of these possible extensions will be computationally expensive; moreover, not all
of the detected dynamic graphlet instances will have meaningful interpretations.

To address this and remove the likely redundant graphlet counts, which is ex-
pected to reduce computational complexity of dynamic graphlet counting, we propose
a modification to the counting process, as follows. When we are extending a dynamic
graphlet ending with event $e_1 = (u_1, v_1, t_1, \sigma_1)$ with a new event $e_2 = (u_2, v_2, t_2, \sigma_2)$,
if $\{u_1, v_1\} = \{u_2, v_2\}$, i.e., if the two events correspond to the same static edge, then
we impose the same two conditions as in the regular counting procedure from Sec-

Intuitively, the new third condition requires a “causal” relationship between $e_1$
and $e_2$: $u_2$ and $v_2$ start their interaction only after the end of $e_1$ (note that there
could still be an event involving $u_2$ and $v_2$ sometime before the start of $e_1$). That is,
in order to extend a dynamic graphlet ending with event $e_1$ with some event $e_2$, the
two nodes participating in $e_2$ should not interact with each other between the start
of $e_1$ and the start of $e_2$, unless $e_1$ and $e_2$ involve the same nodes (otherwise, the
counting process is as in the regular dynamic graphlet procedure). This reduces the number of likely redundant temporal subgraphs that are being considered, which in turn reduces the running time.

We split the counting procedure into the two cases ($e_1$ and $e_2$ being the same static edge vs. different static edges) as we want to impose the new third condition only in the latter case. This is because in the former case we still want to count dynamic graphlet types having consecutive repetitions of the same event, e.g., $D_1$ or $D_4$. And if we imposed the third condition in the former case as well, such a dynamic graphlet type would never be counted.

We refer to this modified counting procedure as constrained dynamic graphlet counting. Note that the only difference in case of constrained dynamic graphlet counting compared to the original counting procedure is which candidate events are chosen to extend a given dynamic graphlet (as determined by procedure $GetNextEvents$ in Appendix Algorithm D.3). In Section 5.2.2 we illustrate the distinction between regular and constrained dynamic graphlet counting procedures. Clearly, constrained dynamic graphlet counting allows for examining fewer instances of a given dynamic graphlet type compared to regular dynamic graphlet counting, because the former excludes from consideration graphlet instances that are likely artifacts of repeated events, unlike the latter. As a consequence, constrained dynamic graphlet counting is expected to be more computationally efficient in terms of running time.

D.3 Experimental Setup

**Graphlet methods under consideration and network construction.** When generating an aggregate or snapshot-based representation of the given temporal network, we began our analysis by testing in detail $w$ values of 1, 2, 3, 5, and 10 on one of our data sets. Since we observed no qualitative differences in results produced by the different choices of this parameter, we continued with the choice of $w = 1$, and we
We also tested multiple values for $t_w$ in each data set, and again we saw no significant qualitative differences in the results. Hence, unless noted otherwise, throughout Section 5.3, we report results for $t_w = 2$ (the unit of time for this parameter depends on the data set).

**Network classification.** The first network evolution model that we use in the context of this analysis was designed to simulate evolution of real-world (social) networks, and it incorporates the following parameters: node arrival rate, initiation of an edge by a node, and selection of edge destination. Specifically, the model is parameterized by the node arrival function $N(t)$ that corresponds to the number of nodes in the network at a given time, parameter $\lambda$ that controls the lifetime of a node, and parameters $\alpha$ and $\beta$ that control how active the nodes are in adding new edges. By choosing different options for the model parameters, we can generate networks with different evolution processes. In particular, for our analysis, we test three different types of the node arrival function (linear, quadratic, and exponential) and two sets of parameters corresponding to edge initiation ($\lambda_1 = 0.032$, $\alpha_1 = 0.8$, $\beta_1 = 0.002$, and $\lambda_2 = 0.02$, $\alpha_2 = 0.9$, $\beta_2 = 0.004$), resulting in six different network classes. We also test a modification of the network evolution model, in which each node upon arrival simply adds a fixed number of edges (in our case, 20) according to preferential attachment and then stops. Intuitively, this modification corresponds to preferential attachment model extended with a node arrival function. In this way, we create three additional network classes, one for each of the three node arrival functions, resulting in nine different network classes in total.

In addition to the above network model originating from social network domain, we perform an equivalent analysis using biological network models. Specifically, we use the following models: geometric gene duplication model with probability cutoff (GEO-GD) and scale-free gene duplication model (SF-GD). GEO-GD starts with a small initial seed network and then iteratively adds new nodes by choosing as the
parent an existing node uniformly at random, and by placing a child node with probability $p$ at a randomly chosen distance of at most $\epsilon$ from the parent and with probability $1 - p$ at a randomly chosen distance of at most $10\epsilon$ from the parent. Here, $\epsilon$ is the same parameter as in the definition of a geometric random graph. Similarly, SF-GD also starts from an initial seed network and then grows it by adding new nodes while relying on principles of gene duplication and mutation. This model has two parameters, $p$ and $q$, corresponding to the probability of the child node keeping interaction with the parent and the probability of the child node forming new connections, respectively. We distinguish between four variations of these two models: GEO-GD with $p = 0.3$, GEO-GD with $p = 0.7$, SF-GD with $p = 0.3$ and $q = 0.7$, and SF-GD with $p = 0.7$ and $q = 0.6$, as these parameter combinations accurately mimic real-world networks. In order to grow the GEO-GD and SF-GD model networks non-uniformly, as in the case with the above social network model, we add a node arrival function on top of GEO-GD and SF-GD. Specifically, we use linear node arrival function for GEO-GD and exponential node arrival function for SF-GD. Note that we intentionally use different node arrival functions for the different biological network models, just as we did with the different classes of the social network model. As with the social network model, we form snapshot-based network representations, test various network sizes (from 1000 to 3000 nodes), and construct 25 networks for each class (Section 5.2.3). Note that here, when we use snapshot-based representation, we report results for $t_w = 5$. We then evaluate graphlet-based methods in the same way as for social network models. To account for at least 90% of variance, here we also need to keep only the first two PCA components.

Node classification. For the network used in this analysis, we report results for the following network construction parameters: $w = 1$ and $t_w = 2$ months. Note that we tested other parameter values as well ($w = 2, 3, 5, 10; t_w = 1$ week, $t_w = 2$ weeks, $t_w = 1$ month, and $t_w = 3$ months), and all results were qualitatively similar.
**Evaluation strategy.** Here, we expand our discussion from Section 5.2.3 on how we measure the PCA performance of a given graphlet approach.

First, we take all possible pairs of objects and retrieve them in the order of increasing distance, starting from the closest ones. We retrieve the object pairs in increments of $k\%$ (including ties), where we vary $k$ from 0% to 100% in increments of 0.01% until we retrieve top 1% of all pairs and in increments of 1% afterwards. If we retrieve a pair with two objects of the same ground truth class, the pair is a true positive, otherwise the pair is a false positive. At a given step, for all pairs that we do not retrieve, the given pair is either a true negative (if it contains objects of different classes) or a false negative (if it contain objects of the same class). Then, at each value of $k$, we compute precision, the fraction of correctly retrieved pairs out of all retrieved pairs, and recall, the fraction of correctly retrieved pairs out of all correct pairs. We find the value of $k$ where precision and recall are equal, and we refer to the resulting precision and recall value as the break-even point. Since lower precision means higher recall, and vice versa, we summarize the two measures into F-score, their harmonic mean, and we report the maximum F-score over all values of $k$. To summarize these results over the whole range of $k$, we measure average method accuracy by computing the area under the precision-recall curve (AUPR). Moreover, we compute an alternative classification accuracy measure, namely the area under the receiver operator characteristic curve (AUROC), which corresponds to the probability of a method ranking a randomly chosen positive pair higher than a randomly chosen negative pair (and so the AUROC value of 0.5 corresponds to a random result). AUPRs are considered to be more credible than AUROCs when there exists imbalance between the size of the set of object pairs that share a class and the size of the set of object pairs that do not share a class.

Second, we split all pairs of objects (i.e., their graphlet-based PCA distances) into two classes: correct pairs (each containing two objects of the same class) and incor-
rect pairs (each containing two objects of two different classes). Then, we compare
distances between correct and incorrect pairs, expecting that distances of the correct
pairs would be statistically significantly lower than distances of the incorrect pairs.
Here, we compare two sets of distances via Wilcoxon rank-sum test.

Finally, for each of the above evaluation tests, we evaluate all graphlet methods
against a random approach. First, as a simple random approach (which favors the
graphlet methods the most), we randomly embed objects into Euclidean space, com-
pute the objects’ pairwise Euclidean distances, and evaluate the resulting random
approach in the same way as above. Second, as a more restrictive random approach
(which favors the graphlet methods the least), for each graphlet method, we keep
its actual PCA distances between objects, and we just randomly permute the object
classes/labels before we evaluate the results. For each randomization approach, we
compute its results as an average over 10 different runs. We report as “random”
approach’s results the highest-scoring values over all of the randomization schemes,
to gain as much confidence as possible into the graphlet approaches’ results.

D.4 Results: the Effect of Graphlet Size

**Number of graphlet nodes.** We test the effect of graphlet size in terms of the num-
ber of nodes on the result quality, for all graphlet methods. For network classification,
increasing the number of graphlet nodes surprisingly leads to inferior accuracy, for all
graphlet methods (Appendix Table D.3). On the other hand, in node classification,
for static and static-temporal graphlets, accuracy is similar for all graphlet sizes, with
4-node graphlets showing marginal superiority, while for dynamic and constrained dy-
namic graphlets, larger number of nodes improves accuracy (Appendix Table D.5).
In terms of the running time, as expected, larger number of nodes increases computa-
tional complexity in all cases.

Note that with node classification, unlike with network classification, the best
parameter version of constrained dynamic graphlet counting is more accurate than the best parameter version of regular dynamic graphlet counting. We note, however, that due to the differences in the counting process, constrained dynamic graphlet counting allows us to consider larger graphlet sizes (e.g., six or seven nodes) that are not attainable when using regular dynamic graphlet counting due to computational constraints (Appendix Table D.5). And it is at these large graphlet sizes of six or seven nodes where constrained dynamic graphlet counting perform the best. So, in order to evaluate which one is more accurate, regular or constrained dynamic graphlet counting, it might not be fair to compare the two methods’ best parameter versions, due to differences in the considered graphlet sizes. Nonetheless, even if we compare regular or constrained dynamic graphlet counting with the same parameters, we find that constrained dynamic graphlet counting still demonstrates better results (Appendix Table D.5).

**Number of graphlet events.** Also, we test the effect of graphlet size in terms of the number of events on the result quality, for dynamic and constrained dynamic graphlets (static and static-temporal graphlets do not deal with events, i.e., temporal edges). For network classification, the number of events does not affect the accuracy (Appendix Table D.3). On the other hand, in node classification, for a fixed number of nodes, the increase in the number of events marginally increases accuracy for dynamic graphlets but decreases accuracy for constrained dynamic graphlets (Appendix Table D.5). In terms of the running time, larger number of events increases computational complexity, although the level of running time increase is less pronounced than when increasing the number of nodes.

**A possible explanation why increasing graphlet size does not always improve accuracy.** Here, we provide a possible explanation behind our observation that increasing graphlet size sometimes leads to decrease in result accuracy in the task of network classification but not in the task of node classification. Namely, in
network classification, we use synthetic networks generated from a *random* graph model, while in node classification we use *real-world* networks. A random graph model might not be able to produce large and dense graphlet structures. Now, given two synthetic networks of different classes, whereas their comparison via smaller graphlets only could correctly identify the networks as dissimilar, their comparison via larger graphlets (with zero counts) as well could mistakenly identify the networks as similar. This is because there are many more larger than smaller graphlets, and thus, the many zero counts for the larger graphlets would match between the networks, wrongly indicating their similarity. However, note that our PCA framework, which reduces graphlet count vectors to a lower dimension (Section 5.2.3), should remove the effect of zero graphlet counts, because it should be preserving only the most relevant graphlet-based information about the network of interest.

An alternative reason why increasing graphlet size (and the number of graphlet nodes in particular) would lead to decrease in performance is as follows. Some graphlet types, such as a claw (e.g., $G_1$ or $G_4$ in Figure 5.1), typically result in an order of magnitude larger counts than other graphlet types, simply because of presence of high-degree (hub) nodes in real-world networks or any scale-free (model) networks. This difference in the magnitude of counts for such highly frequent graphlet types compared to counts of less frequent graphlet types becomes more pronounced as the graphlet size increases. And such extremely dominating graphlet counts could potentially confuse our PCA framework and mistakenly identify the highly frequent graphlet types as more relevant than the less frequent graphlet types. For this reason, one could “rescale” all graphlet counts to get them to the same (or similar) order of magnitude, by, e.g., taking a logarithm of each count [165].

Clearly, determining why increase in graphlet size does not necessarily improve the accuracy of results requires further theoretical and empirical analyses, and this is subject of future work.
Algorithm D.1 Enumerate dynamic graphlet types

1: procedure \textsc{Enumerate}(n_{max}, k_{max})
2: \hspace{1em} \textbf{if} $|V(G)| \leq n_{max}$ \textbf{then}
3: \hspace{2em} $s \leftarrow G'[2k-4]$ \hspace{1em} ▷ recall that $G$ contains $2k - 2$ numbers
4: \hspace{2em} $t \leftarrow G' [2k-3]$
5: \hspace{2em} $u \leftarrow \max(G) + 1$ \hspace{1em} ▷ $u$ is the “new node”
6: \hspace{2em} \textbf{if} $|V(G)| = 2$ \textbf{then} ▷ $V(G)$ is just the set of different numbers in $G$
7: \hspace{3em} \textbf{if} $|V(G)| < n_{max}$ \textbf{then}
8: \hspace{4em} $G_i \leftarrow G + [s, t]$
9: \hspace{4em} \textbf{Graphlets}[k] $\leftarrow$ Graphlets[k] $\cup$ \{G_i\}
10: \hspace{4em} $i \leftarrow i + 1$
11: \hspace{2em} \textbf{end if}
12: \hspace{2em} $G_i \leftarrow G + [t, u]$
13: \hspace{2em} \textbf{Graphlets}[k] $\leftarrow$ Graphlets[k] $\cup$ \{G_i\}
14: \hspace{2em} $i \leftarrow i + 1$
15: \hspace{1em} \textbf{else}
16: \hspace{2em} \textbf{for all} $v \in V(G) - \{s\}$ \textbf{do}
17: \hspace{3em} $G_i \leftarrow G + [\min(s, v), \max(s, v)]$
18: \hspace{3em} \textbf{Graphlets}[k] $\leftarrow$ Graphlets[k] $\cup$ \{G_i\}
19: \hspace{3em} $i \leftarrow i + 1$
20: \hspace{2em} \textbf{end for}
21: \hspace{2em} \textbf{for all} $v \in V(G) - \{s, t\}$ \textbf{do}
22: \hspace{3em} $G_i \leftarrow G + [\min(t, v), \max(t, v)]$
23: \hspace{3em} \textbf{Graphlets}[k] $\leftarrow$ Graphlets[k] $\cup$ \{G_i\}
24: \hspace{3em} $i \leftarrow i + 1$
25: \hspace{2em} \textbf{end for}
26: \hspace{2em} \textbf{if} $|V(G)| < n_{max}$ \textbf{then}
27: \hspace{3em} $G_i \leftarrow G + [s, u]$
28: \hspace{3em} $G_{i+1} \leftarrow G + [t, u]$
29: \hspace{3em} \textbf{Graphlets}[k] $\leftarrow$ Graphlets[k] $\cup$ \{G_i, G_{i+1}\}
30: \hspace{3em} $i \leftarrow i + 2$
31: \hspace{2em} \textbf{end if}
32: \hspace{1em} \textbf{end if}
33: \hspace{1em} \textbf{end if}
34: \hspace{1em} \textbf{end for}
35: \hspace{1em} $k \leftarrow k + 1$
36: \hspace{1em} \textbf{end while}
37: \hspace{1em} \textbf{end procedure}
Algorithm D.2 Counting dynamic graphlets

1:  procedure GraphletCount\((G, n_{max}, k_{max}, \Delta t)\) 
2:     Counts ← dict() 
3:     NCounts ← dict() 
4:     for all \(e \in G.Events()\) do 
5:         GraphletCountFromPrefix\((G, [e], Counts, NCounts, n_{max}, k_{max}, \Delta t)\) 
6:     end for 
7:  end procedure 

Algorithm D.3 Counting dynamic graphlets starting from a prefix

1:  procedure GraphletCountFromPrefix\((G, prefixG, Counts, NCounts, n_{max}, k_{max}, \Delta t)\) 
2:     prefEncoding, prefNodeCodes ← EncodeGraphlet\(\(prefixG\)\) 
3:     Counts[prefEncoding] ← Counts[prefEncoding] + 1 
4:     prefNodes ← prefNodeCodes.keys() 
5:     for all \(v \in \) prefNodes do 
6:         if |\(\text{prefNodes}\)| = 2 then 
7:             nodeCode ← ‘1’ 
8:         else 
9:             nodeCode ← prefNodeCodes[\(v\)] 
10:         end if 
11:     NCounts[\(v\)][prefEncoding+‘,’+nodeCode] ← NCounts[\(v\)][prefEncoding+‘,’+nodeCode] + 1 
12:     end for 
13:     if |\(\text{prefixG}\)| < \(k_{max}\) then 
14:         for all \(e \in \text{getNextEvents}(G, prefixG[[prefixG|−1], \Delta t)\) do 
15:             if |\(\text{prefNodes}\)| < \(n_{max}\) or \(e.EndPoints() \subseteq \text{prefNodes}\) then 
16:                 GraphletCountFromPrefix\((G, \text{prefixG}+ [e], Counts, NCounts, n_{max}, k_{max}, \Delta t)\) 
17:             end if 
18:         end for 
19:     end if 
20:  end procedure
Algorithm D.4 Encoding a dynamic graphlet

```plaintext
1: procedure EncodeGraphlet(Graphlet)
2:     encoding ← 
3:     nodeCodes ← dict()
4:     i ← 1
5:     for all e ∈ Graphlet do
6:         u, v ← e.EndPoints()
7:         updated ← False
8:         if !nodeCodes.hasKey(u) then
9:             nodeCodes[u] ← i
10:            i ← i + 1
11:            updated ← True
12:         end if
13:         if !nodeCodes.hasKey(v) then
14:             nodeCodes[v] ← i
15:            i ← i + 1
16:            updated ← True
17:         end if
18:         if i = 4 and updated = True then
19:             if (nodeCodes[u] = 1 and nodeCodes[v] = 3)
20:                 or (nodeCodes[u] = 3 and nodeCodes[v] = 1) then
21:                 nodeCodes[nodeCodes.GetKey(1)] ← 2
22:                 nodeCodes[nodeCodes.GetKey(2)] ← 1
23:             end if
24:         end if
25:     end for
26:     return encoding, nodeCodes
27: end procedure
```
D.5 Appendix Figures

Figure D.1: Comparison of the different graphlet approaches in the context of biological network classification, in terms of (a) AUPR values, (b) AUROC values, and (c) precision-recall curves.
Figure D.2: Pairwise similarities between the different methods and their parameter variations in the test of node classification. Similarities are computed as Jaccard coefficients between two methods’ top 5% node pairs that are the closest in the graphlet-based PCA space. The order of the methods in the figure directly corresponds to the method order in Appendix Table D.5 (we leave out detailed method names for visual clarity). If we zoom into these results, not only is there a clear separation between static, static-temporal, and (constrained) dynamic graphlets in terms of which nodes they describe as topologically similar, but also, within both dynamic and constrained dynamic graphlets, we can see two clear clusters corresponding to three-node graphlets with different numbers of events. The latter observation suggests that the number of nodes seems to play a larger role in separating the different dynamic graphlet methods compared to the number of events.
Figure D.3: Comparison of the different graphlet approaches in the context of aging-related node classification when considering DyNetAge “ground truth” aging-related data, in terms of (a) AUPR values, (b) AUROC values, and (c) precision-recall curves.

Figure D.4: Comparison of the different graphlet approaches in the context of aging-related node classification when considering BrainExpression2004Age, a non-network “ground truth” aging-related data obtained via gene expression data analyses, in terms of (a) AUPR values, (b) AUROC values, and (c) precision-recall curves.
Figure D.5: Comparison of the different graphlet approaches in the context of aging-related node classification when considering SequenceAge, a non-network “ground truth” aging-related data obtained via genomic sequence analyses, in terms of (a) AUPR values, (b) AUROC values, and (c) precision-recall curves. Note that even though (constrained) dynamic graphlets are not superior with respect to the entire range of $k$ when it comes to SequenceAge, as illustrated in the figure, (constrained) dynamic graphlets are still superior at the lowest $k$ values (Section 5.3.4), as illustrated in Table D.6.
Figure D.6: Overlap between aging-related predictions produced by the four graphlet approaches at (a) the first value of $k$ and (b) the second value of $k$. The total number of predictions in a given set is shown in parentheses under the set’s name.
TABLE D.1

THE NUMBER OF DYNAMIC GRAPHLETS WITH A GIVEN NUMBER OF EVENTS (COLUMNS) THAT HAVE AS THEIR BACKBONE THE SAME STATIC GRAPHLET (ROWS), FOR ALL STATIC GRAPHLETS WITH UP TO FIVE NODES

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$G_1$</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>31</td>
<td>63</td>
<td>127</td>
<td>255</td>
<td>511</td>
</tr>
<tr>
<td>$G_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>25</td>
<td>90</td>
<td>301</td>
<td>966</td>
<td>3,025</td>
<td>9,330</td>
</tr>
<tr>
<td>$G_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>18</td>
<td>56</td>
<td>161</td>
<td>441</td>
<td>1,170</td>
<td>3,036</td>
</tr>
<tr>
<td>$G_4$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>25</td>
<td>90</td>
<td>301</td>
<td>966</td>
<td>3,025</td>
<td>9,330</td>
</tr>
<tr>
<td>$G_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>34</td>
<td>140</td>
<td>525</td>
<td>1,855</td>
<td>6,294</td>
</tr>
<tr>
<td>$G_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>54</td>
<td>320</td>
<td>1,576</td>
<td>7,006</td>
<td>29,238</td>
<td>117,048</td>
</tr>
<tr>
<td>$G_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>150</td>
<td>1,190</td>
<td>7,658</td>
<td>43,690</td>
<td>230,760</td>
</tr>
<tr>
<td>$G_8$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>174</td>
<td>1,842</td>
<td>15,398</td>
<td>111,972</td>
</tr>
<tr>
<td>$G_9$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>26</td>
<td>97</td>
<td>332</td>
<td>1,074</td>
<td>3,339</td>
</tr>
<tr>
<td>$G_{10}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>94</td>
<td>429</td>
<td>1,758</td>
<td>6,735</td>
<td>24,668</td>
</tr>
<tr>
<td>$G_{11}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>65</td>
<td>350</td>
<td>1,701</td>
<td>7,770</td>
<td>34,105</td>
</tr>
<tr>
<td>$G_{12}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>121</td>
<td>926</td>
<td>5,731</td>
<td>31,359</td>
<td>158,475</td>
</tr>
<tr>
<td>$G_{13}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>42</td>
<td>286</td>
<td>1,598</td>
<td>7,968</td>
<td>36,928</td>
</tr>
<tr>
<td>$G_{14}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>120</td>
<td>1,001</td>
<td>6,716</td>
<td>39,747</td>
<td>217,106</td>
</tr>
<tr>
<td>$G_{15}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>43</td>
<td>193</td>
<td>781</td>
<td>2,955</td>
</tr>
<tr>
<td>$G_{16}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>70</td>
<td>456</td>
<td>2,440</td>
<td>11,649</td>
<td>51,648</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>$G_{17}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>62</td>
<td>1,037</td>
<td>10,640</td>
<td>86,462</td>
<td>611,904</td>
</tr>
<tr>
<td>$G_{18}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>109</td>
<td>1,055</td>
<td>8,148</td>
<td>55,084</td>
</tr>
<tr>
<td>$G_{19}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>532</td>
<td>5,104</td>
<td>38,846</td>
<td>257,884</td>
</tr>
<tr>
<td>$G_{20}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>68</td>
<td>573</td>
<td>3,852</td>
<td>22,660</td>
</tr>
<tr>
<td>$G_{21}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>354</td>
<td>3,010</td>
<td>20,522</td>
<td>122,906</td>
</tr>
<tr>
<td>$G_{22}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td>1,063</td>
<td>13,549</td>
<td>134,524</td>
<td></td>
</tr>
<tr>
<td>$G_{23}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>1,954</td>
<td>25,198</td>
<td>251,952</td>
<td></td>
</tr>
<tr>
<td>$G_{24}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>183</td>
<td>3,676</td>
<td>44,367</td>
<td>418,109</td>
<td></td>
</tr>
<tr>
<td>$G_{25}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>69</td>
<td>1,287</td>
<td>14,382</td>
<td>125,419</td>
<td></td>
</tr>
<tr>
<td>$G_{26}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>594</td>
<td>15,442</td>
<td>235,160</td>
<td></td>
</tr>
<tr>
<td>$G_{27}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>219</td>
<td>5,400</td>
<td>77,959</td>
<td></td>
</tr>
<tr>
<td>$G_{28}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1,133</td>
<td>35,859</td>
<td></td>
</tr>
<tr>
<td>$G_{29}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>796</td>
<td></td>
</tr>
</tbody>
</table>
TABLE D.2

THE NUMBER OF DYNAMIC GRAPHLET TYPES WITH UP TO TEN NODES (ROWS) AND TEN EVENTS (COLUMNS)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>13</td>
<td>40</td>
<td>121</td>
<td>364</td>
<td>1,093</td>
<td>3,280</td>
<td>9,841</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>18</td>
<td>116</td>
<td>660</td>
<td>3,542</td>
<td>18,438</td>
<td>94,376</td>
<td>478,440</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>64</td>
<td>680</td>
<td>6,080</td>
<td>49,644</td>
<td>384,384</td>
<td>2,879,440</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>200</td>
<td>3,160</td>
<td>40,600</td>
<td>464,688</td>
<td>4,950,960</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>576</td>
<td>12,656</td>
<td>220,416</td>
<td>3,353,952</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>1,568</td>
<td>45,696</td>
<td>1,034,880</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>4,096</td>
<td>152,832</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>128</td>
<td>10,368</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>256</td>
</tr>
</tbody>
</table>

|    | 1  | 2  | 7  | 36 | 229| 1,678| 13,755| 124,064| 1,217,065| 12,870,970|

* The last row shows cumulative results over all the preceding rows.
<table>
<thead>
<tr>
<th>Method</th>
<th>AUPR</th>
<th>AUROC</th>
<th>Break-even point</th>
<th>Maximum F-score</th>
<th>Running time, s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static, 3-node</td>
<td>0.507</td>
<td>0.935</td>
<td>0.508</td>
<td>0.613</td>
<td>3.3 (1.785)</td>
</tr>
<tr>
<td>Static, 4-node</td>
<td>0.423</td>
<td>0.882</td>
<td>0.463</td>
<td>0.468</td>
<td>3.3 (1.785)</td>
</tr>
<tr>
<td>Static, 5-node</td>
<td>0.321</td>
<td>0.807</td>
<td>0.341</td>
<td>0.376</td>
<td>3.3 (1.785)</td>
</tr>
<tr>
<td>Static-temporal, 3-node</td>
<td>0.784</td>
<td>0.947</td>
<td>0.702</td>
<td>0.707</td>
<td>3.7 (0.173)</td>
</tr>
<tr>
<td>Static-temporal, 4-node</td>
<td>0.498</td>
<td>0.826</td>
<td>0.475</td>
<td>0.476</td>
<td>3.7 (0.173)</td>
</tr>
<tr>
<td>Static-temporal, 5-node</td>
<td>0.374</td>
<td>0.790</td>
<td>0.379</td>
<td>0.390</td>
<td>3.7 (0.173)</td>
</tr>
<tr>
<td>Dynamic, 3-event, 3-node</td>
<td>0.960</td>
<td>0.994</td>
<td>0.884</td>
<td>0.885</td>
<td>0.6 (0.116)</td>
</tr>
<tr>
<td>Dynamic, 5-event, 3-node</td>
<td>0.960</td>
<td>0.994</td>
<td>0.884</td>
<td>0.885</td>
<td>0.7 (0.104)</td>
</tr>
<tr>
<td>Dynamic, 7-event, 3-node</td>
<td>0.960</td>
<td>0.994</td>
<td>0.884</td>
<td>0.885</td>
<td>1.4 (0.149)</td>
</tr>
<tr>
<td>Dynamic, 6-event, 4-node</td>
<td>0.714</td>
<td>0.937</td>
<td>0.656</td>
<td>0.660</td>
<td>4.8 (0.875)</td>
</tr>
<tr>
<td>Constrained dynamic, 3-event, 3-node</td>
<td>0.949</td>
<td>0.993</td>
<td>0.881</td>
<td>0.881</td>
<td>0.6 (0.188)</td>
</tr>
<tr>
<td>Constrained dynamic, 5-event, 3-node</td>
<td>0.949</td>
<td>0.993</td>
<td>0.881</td>
<td>0.881</td>
<td>0.7 (0.145)</td>
</tr>
<tr>
<td>Constrained dynamic, 7-event, 3-node</td>
<td>0.949</td>
<td>0.993</td>
<td>0.881</td>
<td>0.881</td>
<td>1.5 (0.206)</td>
</tr>
<tr>
<td>Constrained dynamic, 6-event, 4-node</td>
<td>0.740</td>
<td>0.939</td>
<td>0.672</td>
<td>0.675</td>
<td>4.2 (0.684)</td>
</tr>
</tbody>
</table>

*Different columns correspond to different performance measures. In a given column, the value in bold corresponds to the best result over all methods. Numbers in parentheses correspond to standard deviations. For illustration purposes, graphlet counting running times are shown for one of the nine network classes (using the exponential node addition function and the first set of edge initiation parameters (Appendix Section D.3)); running times for the remaining network classes are shown in Appendix Table D.4. Note that for static and static-temporal graphlets, running times for 3- and 4-node graphlets are the same as for 5-node graphlets simply because their implementations compute graphlet counts for all up to 5-node graphlets by default and then they compute graphlet counts for smaller graphlet sizes simply by removing counts corresponding to the larger graphlet sizes.
<table>
<thead>
<tr>
<th>Method</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type 1</td>
<td>Type 2</td>
<td>PA</td>
</tr>
<tr>
<td>Static, 3-node</td>
<td>5.4 (2.914)</td>
<td>18.1 (4.957)</td>
<td>1316.2 (131.210)</td>
</tr>
<tr>
<td>Static, 4-node</td>
<td>5.4 (2.914)</td>
<td>18.1 (4.957)</td>
<td>1316.2 (131.210)</td>
</tr>
<tr>
<td>Static, 5-node</td>
<td>5.4 (2.914)</td>
<td>18.1 (4.957)</td>
<td>1316.2 (131.210)</td>
</tr>
<tr>
<td>Static-temporal, 3-node</td>
<td>47.7 (0.710)</td>
<td>49.1 (0.847)</td>
<td>48.5 (1.011)</td>
</tr>
<tr>
<td>Static-temporal, 4-node</td>
<td>47.7 (0.710)</td>
<td>49.1 (0.847)</td>
<td>48.5 (1.011)</td>
</tr>
<tr>
<td>Static-temporal, 5-node</td>
<td>47.7 (0.710)</td>
<td>49.1 (0.847)</td>
<td>48.5 (1.011)</td>
</tr>
<tr>
<td>Dynamic, 3-event, 3-node</td>
<td>0.8 (0.153)</td>
<td>1.2 (0.317)</td>
<td>2.4 (0.190)</td>
</tr>
<tr>
<td>Dynamic, 5-event, 3-node</td>
<td>0.9 (0.288)</td>
<td>1.4 (0.297)</td>
<td>2.5 (0.365)</td>
</tr>
<tr>
<td>Dynamic, 7-event, 3-node</td>
<td>1.6 (0.262)</td>
<td>2.2 (0.322)</td>
<td>3.3 (0.353)</td>
</tr>
<tr>
<td>Dynamic, 6-event, 4-node</td>
<td>3.4 (0.530)</td>
<td>4.8 (0.416)</td>
<td>5.9 (0.521)</td>
</tr>
<tr>
<td>Constrained dynamic, 3-event, 3-node</td>
<td>0.8 (0.224)</td>
<td>1.1 (0.207)</td>
<td>2.7 (0.341)</td>
</tr>
<tr>
<td>Constrained dynamic, 5-event, 3-node</td>
<td>0.9 (0.206)</td>
<td>1.3 (0.241)</td>
<td>2.7 (0.228)</td>
</tr>
<tr>
<td>Constrained dynamic, 7-event, 3-node</td>
<td>1.6 (0.205)</td>
<td>2.1 (0.302)</td>
<td>3.5 (0.372)</td>
</tr>
<tr>
<td>Constrained dynamic, 6-event, 4-node</td>
<td>3.3 (0.491)</td>
<td>4.4 (0.540)</td>
<td>6.1 (0.566)</td>
</tr>
</tbody>
</table>

* The first header row lists three possible node arrival functions. The second header row lists three possible edge initiation strategies. The two combined result in $3 \times 3 = 9$ network classes.
TABLE D.5

DETAILED NODE CLASSIFICATION RESULTS FOR THE
DIFFERENT METHODS AND DIFFERENT PARAMETERS IN EACH
METHOD

<table>
<thead>
<tr>
<th>Method</th>
<th>AUPR</th>
<th>AUROC</th>
<th>Break-even point</th>
<th>Maximum F-score</th>
<th>Running time, s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static, 3-node</td>
<td>0.464</td>
<td>0.600</td>
<td>0.456</td>
<td>0.562</td>
<td>9.4</td>
</tr>
<tr>
<td>Static, 4-node</td>
<td>0.469</td>
<td>0.610</td>
<td>0.461</td>
<td>0.567</td>
<td>9.4</td>
</tr>
<tr>
<td>Static, 5-node</td>
<td>0.464</td>
<td>0.604</td>
<td>0.462</td>
<td>0.566</td>
<td>9.4</td>
</tr>
<tr>
<td>Static-temporal, 3-node</td>
<td>0.499</td>
<td>0.644</td>
<td>0.508</td>
<td>0.571</td>
<td>2.7</td>
</tr>
<tr>
<td>Static-temporal, 4-node</td>
<td>0.503</td>
<td>0.643</td>
<td>0.609</td>
<td>0.689</td>
<td>2.7</td>
</tr>
<tr>
<td>Static-temporal, 5-node</td>
<td>0.482</td>
<td>0.570</td>
<td>0.486</td>
<td>0.554</td>
<td>2.7</td>
</tr>
<tr>
<td>Dynamic, 3-event, 3-node</td>
<td>0.479</td>
<td>0.622</td>
<td>0.477</td>
<td>0.569</td>
<td>2.7</td>
</tr>
<tr>
<td>Dynamic, 5-event, 3-node</td>
<td>0.474</td>
<td>0.615</td>
<td>0.458</td>
<td>0.569</td>
<td>9.6</td>
</tr>
<tr>
<td>Dynamic, 7-event, 3-node</td>
<td>0.470</td>
<td>0.609</td>
<td>0.460</td>
<td>0.572</td>
<td>27.5</td>
</tr>
<tr>
<td>Dynamic, 3-event, 4-node</td>
<td>0.541</td>
<td>0.684</td>
<td>0.547</td>
<td>0.594</td>
<td>24.5</td>
</tr>
<tr>
<td>Dynamic, 6-event, 4-node</td>
<td>0.525</td>
<td>0.666</td>
<td>0.516</td>
<td>0.583</td>
<td>1,024</td>
</tr>
<tr>
<td>Dynamic, 4-event, 5-node</td>
<td>0.591</td>
<td>0.726</td>
<td>0.615</td>
<td>0.620</td>
<td>753</td>
</tr>
<tr>
<td>Constrained dynamic, 3-event, 3-node</td>
<td>0.491</td>
<td>0.639</td>
<td>0.498</td>
<td>0.569</td>
<td>1.1</td>
</tr>
<tr>
<td>Constrained dynamic, 5-event, 3-node</td>
<td>0.492</td>
<td>0.638</td>
<td>0.495</td>
<td>0.570</td>
<td>1.9</td>
</tr>
<tr>
<td>Constrained dynamic, 7-event, 3-node</td>
<td>0.492</td>
<td>0.638</td>
<td>0.495</td>
<td>0.571</td>
<td>2.6</td>
</tr>
<tr>
<td>Constrained dynamic, 3-event, 4-node</td>
<td>0.550</td>
<td>0.695</td>
<td>0.570</td>
<td>0.600</td>
<td>4.9</td>
</tr>
<tr>
<td>Constrained dynamic, 6-event, 4-node</td>
<td>0.550</td>
<td>0.695</td>
<td>0.571</td>
<td>0.600</td>
<td>37.2</td>
</tr>
<tr>
<td>Constrained dynamic, 4-event, 5-node</td>
<td>0.594</td>
<td>0.732</td>
<td>0.618</td>
<td>0.637</td>
<td>60.8</td>
</tr>
<tr>
<td>Constrained dynamic, 5-event, 6-node</td>
<td><strong>0.611</strong></td>
<td><strong>0.743</strong></td>
<td><strong>0.636</strong></td>
<td><strong>0.654</strong></td>
<td>815</td>
</tr>
<tr>
<td>Constrained dynamic, 6-event, 7-node</td>
<td>0.608</td>
<td>0.742</td>
<td>0.635</td>
<td>0.652</td>
<td>10,029</td>
</tr>
</tbody>
</table>

Random                      | 0.376 (0.009) | 0.495 (0.016) | 0.369 (0.007) | 0.550 (0.000) | -               |

*The table can be interpreted just as Appendix Table D.3. Notice that in this test of node classification we could test some additional parameters (e.g., graphlets on five or more nodes) compared to the test of network classification (Appendix Table D.3); this is because the test of network classification is computationally much more complex, given that graphlets need to be counted in multiple networks, as opposed to counting graphlets in only one network in the node classification task.*
TABLE D.6

PRECISION OF THE DIFFERENT METHODS IN THE CONTEXT OF AGING AT THE TWO $k$ VALUES, FOR THE THREE “GROUND TRUTH” AGING-RELATED DATA SETS (DyNetAge, BrainExpression2004Age, AND SequenceAge)

<table>
<thead>
<tr>
<th></th>
<th>DyNetAge</th>
<th>BrainExpression2004Age</th>
<th>SequenceAge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k_1$</td>
<td>$k_2$</td>
<td>$k_1$</td>
</tr>
<tr>
<td>Static</td>
<td>0.981</td>
<td>0.981</td>
<td>0.947</td>
</tr>
<tr>
<td>Static-temporal</td>
<td>0.992</td>
<td>0.992</td>
<td><strong>0.948</strong></td>
</tr>
<tr>
<td>Dynamic</td>
<td><strong>0.998</strong></td>
<td><strong>0.998</strong></td>
<td>0.926</td>
</tr>
<tr>
<td>Constrained dynamic</td>
<td>0.993</td>
<td>0.993</td>
<td>0.927</td>
</tr>
<tr>
<td>Random</td>
<td>0.850</td>
<td>0.851</td>
<td>0.946</td>
</tr>
</tbody>
</table>

* For each method, the highest-scoring graphlet size is chosen. In a column, the value in bold is the best result over all methods. Note that even though (constrained) dynamic graphlets are not superior with respect to the two values of $k$ when it comes to BrainExpression2004Age, as illustrated in the table, (constrained) dynamic graphlets are still superior with respect to the entire range of $k$, as illustrated in Appendix Figure D.4.
TABLE D.7

A SAMPLE OF THE LIST OF AGING-RELATED PREDICTIONS PRODUCED BY THE FOUR GRAPHLET APPROACHES AT THE TWO VALUES OF \( k \)

<table>
<thead>
<tr>
<th>Gene name</th>
<th>Gene ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBWD2</td>
<td>150472</td>
</tr>
<tr>
<td>GBE1</td>
<td>2632</td>
</tr>
<tr>
<td>ELL2</td>
<td>22936</td>
</tr>
<tr>
<td>PNKD</td>
<td>25953</td>
</tr>
<tr>
<td>GSTM3</td>
<td>2947</td>
</tr>
</tbody>
</table>

* Note that this is just a sample (top five rows for dynamic graphlets at the first value of \( k \)) of the entire table that is available online at [http://www.nd.edu/~cone/DG/predictions.xlsx](http://www.nd.edu/~cone/DG/predictions.xlsx). We provide a sample only since the entire table is too large to be directly incorporated into the Appendix. Note that the parameter versions of the graphlet approaches that were used to make the predictions are not necessarily the same the parameter versions shown in Table 5.1. Namely, the dynamic graphlet approach at \( k_1 \), which had precision of 0.983 in Table 5.1, resulted in only a single novel aging-related gene, and as such, we had no statistical power to validate this version of the dynamic graphlet approach. Thus, to produce novel aging-related predictions, we used the next highest scoring (in terms of precision) version of the dynamic graphlet approach. Also, we note that one parameter version of static-temporal graphlets produced the maximum precision of 1 with respect to the known aging-related knowledge from DyNetAge for the first value of \( k \). This is interesting, as it means that its predictions are a perfect subset of DyNetAge, despite the two approaches having significant differences: our static-temporal graphlet approach from this study is based on ranking of node pairs and on graphlets only, whereas the DyNetAge study is based on the notion of changing node centralities and on multiple measures of network topology. However, whereas this parameter version of the static-temporal approach resulted in the perfect precision, it has no practical usefulness beyond predicting knowledge that is already predicted by DyNetAge; it cannot predict any novel aging-related genes. For this reason, we left out this version from consideration in our study, and instead we have focused on the next highest scoring version of the static-temporal approach, from which we have made novel aging-related predictions.
TABLE D.8

GO TERMS ENRICHED IN NOVEL AGING-RELATED PREDICTIONS OF DYNAMIC GRAPHLETS AT THE SECOND VALUE OF $k$

<table>
<thead>
<tr>
<th>GO term ID</th>
<th>GO term name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0044281</td>
<td>Small molecule metabolic process</td>
</tr>
<tr>
<td>GO:0046854</td>
<td>Phosphatidylinositol phosphorylation</td>
</tr>
<tr>
<td>GO:0046677</td>
<td>Response to antibiotic</td>
</tr>
<tr>
<td>GO:0050885</td>
<td>Neuromuscular process controlling balance</td>
</tr>
<tr>
<td>GO:0071356</td>
<td>Cellular response to tumor necrosis factor</td>
</tr>
<tr>
<td>GO:0006661</td>
<td>Phosphatidylinositol biosynthetic process</td>
</tr>
<tr>
<td>GO:0001666</td>
<td>Response to hypoxia</td>
</tr>
<tr>
<td>GO:007059</td>
<td>Chromosome segregation</td>
</tr>
<tr>
<td>GO:0048565</td>
<td>Digestive tract development</td>
</tr>
<tr>
<td>GO:0010043</td>
<td>Response to zinc ion</td>
</tr>
<tr>
<td>GO:0071320</td>
<td>Cellular response to cAMP</td>
</tr>
<tr>
<td>GO:0006364</td>
<td>rRNA processing</td>
</tr>
<tr>
<td>GO:0006805</td>
<td>Xenobiotic metabolic process</td>
</tr>
<tr>
<td>GO:0044237</td>
<td>Cellular metabolic process</td>
</tr>
<tr>
<td>GO:0044267</td>
<td>Cellular protein metabolic process</td>
</tr>
<tr>
<td>GO:0016337</td>
<td>Single organismal cell-cell adhesion</td>
</tr>
<tr>
<td>GO:0019886</td>
<td>Antigen processing and presentation of exogenous peptide antigen via MHC class II</td>
</tr>
</tbody>
</table>

* GO terms that are also enriched in SequenceAge are shown in bold. All other GO terms are enriched only in dynamic graphlets' predictions.
E.1 Related Work

Here, we expand our discussion from Section 6.1.2 and discuss the three existing methods, GraphScope, Multi-Step, and GHRG, which can deal with the SCD problem.

**GraphScope** [199] works as follows. The first snapshot becomes the current segment. Given the current segment, the method iteratively examines the next snapshot in the temporal sequence to determine whether: 1) the community organization of the snapshot in question matches well the community organization of the current segment, and thus, the snapshot should be added to the current segment (this simply extends the current segment for the next iteration), or instead 2) the community organization of the snapshot does not match well the community organization of the current segment, and thus, the snapshot should begin a new segment (which becomes the current segment for the next iteration). Community organizations of the given snapshot and the current segment are obtained and their match is measured via the minimum description length (MDL) principle.

**Multi-Step** [111] uses an agglomerative hierarchical clustering approach as follows. Each snapshot starts as a singleton segment. Then, in every iteration, the most similar (in terms of community organization) pair of segments are combined. Specifically, the level of similarity between two segments quantifies how well the community organization (i.e., the partition) of the first segment fits the second segment,
and also how well the partition of the second segment fits the first segment. Here, the quality of the fit of a partition to a segment is based on average modularity (Appendix Section E.2.1), and a partition for the given segment is detected by greedily maximizing average modularity via a modification of Louvain algorithm for static community detection (Appendix Section E.2.2). The output of the above iterative Multi-Step procedure is a hierarchical tree with snapshots as leaves. However, it is not clear how to automatically cut the tree to obtain segments and their corresponding change points. As such, Multi-Step is suitable when the desired number of segments is provided as input.

**GHRG** [156] considers a fixed-length sliding window of the most recent snapshots and uses a statistical test to evaluate whether: 1) within the window, the snapshots before and after a given time point originate from different community organization-related models, and thus, this time point should be declared as a change point, or instead 2) all snapshots within the window come from the same model, and thus, there is no change point in that window. As its community organization-related model, GHRG uses generalized hierarchical random graphs.

### E.2 Our SCOUT Approach

Here, we expand our discussion from Section 6.2.3 on the three main components of SCOUT: objective function (Appendix Section E.2.1), consensus clustering (Appendix Section E.2.2), and search strategy (Appendix Section E.2.3).

#### E.2.1 Objective Function

For the CSCD problem, in which segmentation parsimony is fixed, an objective function $Q$ should measure partition accuracy of an output $O$. For the SCD problem, an objective function $Q$ should measure both segmentation parsimony and partition accuracy. We organize the rest of this section as follows. I) We discuss the group
of objective functions $Q_P$ that measure only partition accuracy. II) We discuss the group of objective functions $Q_B$ that measure both aspects of the output quality. III) We discuss how to use the above two groups of objective functions to solve the CSCD and SCD problems.

I) To measure only partition accuracy, we define $Q_P$ as the average snapshot partition quality:

$$Q_P(O, D) = \frac{1}{k} \sum_{i=0}^{l-1} \sum_{j=t_i}^{t_{i+1}-1} F(p_i, G_j),$$  \hspace{1cm} (E.1)

where $F$ measures the fit of partition $p_i$ to snapshot $G_j$. Since there is no one universally accepted measure $F$ of how well a given partition $p$ fits a given snapshot $G = (V, E)$, we test four popular such measures $F$ [221]. Let $|p|$ be the number of clusters in partition $p$. For a given cluster $c \in p$, let $n_c$ be the number of its nodes, let $m_c$ be the number of its internal edges, and let $b_c$ be the number of its boundary edges (edges between the nodes in $c$ and the nodes in $V \not\subset c$). We consider the following choices of $F$: 1) Modularity [149]: $F(p, G) = \frac{1}{2|E|} \sum_{c \in p} (m_c - E(m_c))$, where $E(m_c)$ is the expected number of $c$'s internal edges under a configuration model (a random model with the same degree distribution as $G$). Intuitively, a partition is of high quality with respect to modularity if its clusters are denser than at random. The higher the modularity score, the better the partition accuracy. The remaining three measures are based on the intuition that in a good partition, clusters should have more inside than boundary edges. 2) Conductance [31]: $F(p, G) = \frac{1}{|p|} \sum_{c \in p} b_c$, where $b_c$ is the number of boundary edges of $c$. 3) Normalized Cut [188]: $F(p, G) = \frac{1}{|p|} \sum_{c \in p} \frac{b_c}{2m_c+n_c} + \frac{b_c}{2(m-m_c)+n_c}$. 4) Average-ODF [60]: $F(p, G) = \frac{1}{|p|} \sum_{c \in p} \frac{1}{|n_c|} \sum_{u \in c} \frac{\left|\{v \in E : v \not\in c\}\right|}{d_u}$, where $d_u$ is the degree of node $u$. Because for the last three measures, the lower the score, the better the partition accuracy, and because SCOUT aims to maximize (rather than minimize) its objective function, SCOUT uses $F'(p, G) = 1 - F(p, G)$ instead of $F$ in its objective function for these three measures.
II) To simultaneously measure both segmentation parsimony and partition accuracy, we define $Q_B$ based on the *model selection problem* [89]. Intuitively, given some $O = (T, P)$ for a dynamic network $D$, if we use $O$ as a generative model for creating a dynamic network, how well does this model $O$ fit $D$? On the one hand, the more complex the model (intuitively, the more segments there are in $O$, i.e., the lower the segmentation parsimony, and also, the more clusters there are in each segment partition), the more likely it is that we will observe a high fit (as measured by the likelihood of $D$ given $O$, which mostly reflects partition accuracy). On the other hand, the less complex the model, the more likely it is that we will observe a low fit. Given a set of $O$s under consideration (see below), the goal of the model selection problem is to choose $O^*$ that optimizes some measure of quality over all such $O$s. This measure of quality should balance between the *goodness of the fit of $O$ to $D* (mostly partition accuracy) and the *complexity of the model $O* (mostly segmentation parsimony).

To solve the model selection problem, we test two popular approaches [70, 89]: 1) *Akaike Information Criterion* (AIC) [4] and 2) *Bayesian Information Criterion* (BIC) [183]. Both approaches compute the goodness of the fit in the same way. They also compute the complexity of the model in the same way. However, the two approaches differ in how they penalize the objective function by the complexity of the model. We define $Q_B$ using AIC or BIC as follows:

$$Q_B(O, D) = \ell(D|O) - w(D)N_p(O).$$  \hspace{1cm} (E.2)

In the above formula, the goodness of the fit is measured via $\ell(D|O)$, the log-likelihood of $D$ given $O$ (see below). The complexity of the model is measured via $N_p(O)$, the *number of parameters* in $O$ (see below). The above two quantities, $\ell(D|O)$ and $N_p(O)$, are balanced via *penalty weight* $w(D)$. For AIC, $w(D) = 1$. For BIC, $w(D) = \ldots$
\( \log N_0(D) \), where \( N_0(D) \) is the number of observations in \( D \) (how “large” \( D \) is; see below). In general, \( w(D) \) is larger in BIC than in AIC, which means that BIC penalizes complex models more heavily than AIC. Intuitively, in our case, this means that BIC prefers outputs with smaller numbers of segments than AIC.

Next, we discuss how to compute \( \ell(D|O) \), \( N_p(O) \), and \( N_0(D) \).

To compute \( \ell(D|O) \), we assume that each segment \( s_i \) is independent of the others, and thus \( \ell(D|O) \) is just the sum of log-likelihoods of the individual segments \( \ell(s_i|O) \):

\[
\ell(D|O) = \sum_{i=0}^{l-1} \ell(s_i|O), \tag{E.3}
\]

where segmentation \( S = \{s_0, s_1, \ldots, s_l\} \) is determined by change point set \( T \) of \( O \).

To compute \( \ell(s_i|O) \), we assume that \( s_i \) has an associated stochastic blockmodel (see below) and each snapshot \( G_j \) within segment \( s_i \) is independent given this blockmodel. A stochastic blockmodel is a generative model where probability of an edge is determined by the cluster memberships of its endpoints [90]. The blockmodel contains two parts: a partition \( p \) and a stochastic block matrix \( \theta \) of size \(|p| \times |p|\), where \( \theta_{c_u,c_v} \) is the probability of an edge between two nodes \( u, v \) from clusters \( c_u, c_v \in p \), respectively.

The blockmodel associated with \( s_i \) is based on the corresponding segment partition \( p_i \) and has the stochastic block matrix \( \hat{\theta}^{(i)} \) (see below). Thus, \( \ell(s_i|O) \) is just the sum of log-likelihoods of the individual snapshots \( \ell(G_j|\hat{\theta}^{(i)}, p_i) \):

\[
\ell(s_i|O) = \sum_{j=t_i}^{t_{i+1}-1} \ell(G_j|\hat{\theta}^{(i)}, p_i), \tag{E.4}
\]

where \( \hat{\theta}^{(i)} \) is the maximum likelihood estimator of \( \theta^{(i)}_{c_u,c_v} \). That is, \( \hat{\theta}^{(i)}_{c_u,c_v} \) is computed as the fraction of the actual and the maximum possible numbers of edges between
nodes in cluster $c_u$ and nodes in cluster $c_v$ across all snapshots $G_j$ of segment $s_i$:

$$\hat{\theta}^{(i)}_{c_u,c_v} = \frac{\sum_{j=t_i}^{t_{i+1}-1} m^{(j)}_{c_u c_v}}{\sum_{j=t_i}^{t_{i+1}-1} n^{(j)}_{c_u c_v}}. \quad (E.5)$$

In the above formula, $m^{(j)}_{c_u c_v}$ is the number of edges in $G_j$ between nodes in cluster $c_u$ and nodes in cluster $c_v$, and $n^{(j)}_{c_u c_v}$ is the maximum possible number of such edges. If $c_u \neq c_v$, then $n^{(j)}_{c_u c_v} = n^{(j)}_{c_u} n^{(j)}_{c_v}$, where $n^{(j)}_{c_u}$ and $n^{(j)}_{c_v}$ are the numbers of nodes from $G_j$ that are in clusters $c_u$ and $c_v$, respectively. If $c_u = c_v$, then $n^{(j)}_{c_u c_v} = \left(\frac{n^{(j)}_{c_u}}{2}\right)$. To compute $\ell(G_j|\hat{\theta}^{(i)}, p_i)$, the log-likelihood of $G_j = (V_j, E_j)$ given $\hat{\theta}^{(i)}$ and $p_i$, because we are using a stochastic blockmodel, we assume that an edge between each pair of nodes $u, v \in V_j$ is independent of others and its probability is based on the cluster memberships $c_u, c_v \in p_i$ of $u, v$, respectively. Thus, $\ell(G_j|\hat{\theta}^{(i)}, p_i)$ is just the sum of log-likelihoods of individual edges and non-edges observed in $G_j$:

$$\ell(G_j|\hat{\theta}^{(i)}, p_i) = \sum_{(u,v) \in E_j} \log \hat{\theta}^{(i)}_{c_u,c_v} + \sum_{(u,v) \notin E_j} \log(1 - \hat{\theta}^{(i)}_{c_u,c_v}). \quad (E.6)$$

To compute $N_p(O)$, we count the number of values in $\hat{\theta}^{(i)}$s across all segments $s_i, i \in [0, l-1]$. For a given segment $s_i$, we have one value in $\hat{\theta}^{(i)}$ for each pair of clusters in $p_i$ (including a cluster with itself), so, in total:

$$N_p(O) = \sum_{i=0}^{l-1} \left(\left\lfloor \frac{|p_i|}{2} \right\rfloor + |p_i| \right) = \sum_{i=0}^{l-1} |p_i| \left(\left\lfloor \frac{|p_i|}{2} \right\rfloor + 1 \right). \quad (E.7)$$

To compute $N_o(D)$, we count the number of node pairs in all snapshots $G_j = (V_j, E_j)$ in $D$ (Appendix Equation [E.6]):

$$N_o(D) = \sum_{j=0}^{k-1} \left(\left\lfloor \frac{|V_j|}{2} \right\rfloor \right) = \sum_{j=0}^{k-1} |V_j| \left(\left\lfloor \frac{|V_j|}{2} \right\rfloor - 1 \right). \quad (E.8)$$

By combining Appendix Equations [E.3], [E.7] and, for BIC, [E.8] we can compute
$Q_B(O, D)$ in Appendix Equation $E.2$

**III** Given some consensus clustering method and search strategy (see below), and given the above two groups of objective functions, $Q_P$ and $Q_B$, we now discuss how to solve the CSCD and SCD problems.

**To solve the CSCD problem,** we pick as $O^*$ a solution with the desired number of segments $l$ that maximizes $Q \in \{Q_P, Q_B\}$:

$$O^* = \arg \max_{|T| = l - 1, O \in R} Q(O, D), \quad (E.9)$$

where $T$ is the change point set of $O$ (recall that we need $l - 1$ change points to produce $l$ segments) and $R$ is the set of the considered outputs (note that this set is determined by the search strategy; see below). Here, $Q$ can measure either only partition accuracy (i.e., $Q_P$) or both aspects of the SCD problem (i.e., $Q_B$).

**To solve the SCD problem,** we first solve the CSCD problem $\forall l \in [1, k]$ using $Q_P$ or $Q_B$ as described above, and then we pick as $O^*$ one of these $k$ solutions that maximizes $Q_B$. Let $R^* = \{O^*_i | i \in [1, k]\}$, where $O^*_i$ is the solution of the CSCD problem with $i$ segments (Appendix Equation $E.9$). Given $R^*$, we select $O^*$ as follows:

$$O^* = \arg \max_{O \in R^*} Q_B(O, D). \quad (E.10)$$

Note that if we use the same $Q_B$ when constructing $R^*$ (Appendix Equation $E.9$) and when selecting $O^*$ from $R^*$ (Appendix Equation $E.10$), the described procedure for solving the SCD problem is equivalent to directly aiming to find $O^*$ with the optimal value of $Q_B$.

E.2.2 Consensus Clustering

Given change point set $T$, we obtain the set of segment partitions $P$ by applying consensus clustering to each segment. That is, for each segment $s_i$, we aim to find a
single partition $p_i$ that works well for all snapshots in $s_i$. Note that if $s_i$ contains only one snapshot, consensus clustering is equivalent to simple static network clustering, since there are no multiple snapshots to compute consensus for; yet, for consistency, we still refer to such clustering process as consensus clustering. Intuitively, the chosen consensus clustering method should align with the objective function, meaning that, for a given change point set $T$, consensus clustering should aim to find the set $P$ of segment partitions that maximize the objective function $Q$. We consider three consensus clustering methods: *sum graph* [11], *Average-Louvain* [11], and *consensus matrix* [106].

**Sum graph.** An intuitive way to perform consensus clustering for a given segment $s_i$ is to first construct a special graph that “summarizes” the topology of all snapshots in $s_i$ and then find community organization in this “summary” graph under the hypothesis that this organization will fit well all snapshots in $s_i$. Here, we construct this “summary” graph for $s_i$ simply as a *sum graph*, a weighted graph whose adjacency matrix is the sum of the adjacency matrices of all snapshots in $s_i$ [11]. Then, we use a static community detection method that can handle weighted graphs to find a partition in this sum graph. We test seven popular static community detection methods [61]: 1) *Fast Modularity* [36]: the method starts with each node as a singleton community, and then at every iteration it merges two communities to greedily optimize modularity. 2) *Label Propagation* [170]: the method starts with each node as a singleton community (referred to as a *label*), and then at every iteration each node adopts the label used by the majority of its neighbors. 3) *Leading Eigenvector* [148]: the method optimizes modularity based on the eigenspectrum of a modularity matrix (a matrix analogous to graph Laplacian in graph partitioning). 4) *Infomap* [177]: the method aims to find a partition minimizing the expected description length of a random walker trajectory. 5) *Walktrap* [162]: the method finds a partition based on the intuition that short random walks tend to get “trapped” in the same com-
munity, since, intuitively, there are many edges pointing inside the community and only few pointing outside. 6) Louvain [22]: the method starts with each node as a singleton community and then repeatedly performs two phases: greedily optimizing modularity by moving nodes to neighboring communities and constructing a new graph with communities as nodes. 7) Stabilized Louvain [10]: a modification of Louvain algorithm for snapshot clustering that aims to produce stable partitions (i.e., prevent two snapshots with similar topologies from having dissimilar partitions); to achieve stability, the method clusters a snapshot at time $t$ via Louvain algorithm initialized with the partition obtained for the snapshot at time $t - 1$.

**Average-Louvain.** This method aims to find a segment partition $p_i$ that maximizes average modularity over all snapshots in $s_i$ [11]. To achieve this, the method uses a modification of Louvain algorithm for static community detection (see above). Recall that Louvain method contains two phases. In Average-Louvain, the first phase is modified so that the modularity gain of each move is computed as the average gain of this move across all snapshots in the given segment. The second phase, constructing a network of communities, is modified so that the same transformation is performed independently on all snapshots within the given segment. Thus, all snapshots have the same partition, which becomes $p_i$.

**Consensus matrix.** This method aims to find a segment partition $p_i$ directly from the partitions of snapshots in $s_i$ [106]. That is, given individual snapshot partitions as input, the method computes a *consensus matrix* $M$ based on the co-occurrence of nodes in clusters of the input partitions. Specifically, entry $M_{ij}$ of this matrix indicates the fraction of the input partitions in which nodes $i$ and $j$ are in the same cluster. Matrix $M$, which can be thought of as a weighted graph, can then be clustered by some static community detection method to produce a consensus partition. To compute snapshot partitions as well as to cluster $M$, we use the same static community detection methods as for the sum graph approach above.
E.2.3 Search Strategy

We test three strategies for exploring the space of possible change point sets: the exhaustive search, top-down search, and bottom-up search. Each strategy first produces one best solution for each possible number of segments for the CSCD problem (Appendix Equation E.9), which are then used to solve the SCD problem (Appendix Equation E.10). The first strategy is aimed at producing a globally optimal solution at the expense of larger running time, while the last two are heuristics aimed at producing a good solution in a faster manner. Below, for each strategy, we discuss how the strategy works and its “conceptual” computational complexity. By “conceptual”, we mean that we express the running time of a given strategy in terms of the number of times that consensus clustering is performed. We do this because: 1) performing consensus clustering is SCOUT’s most computationally intensive step whose running time dominates all other steps, and 2) we vary consensus clustering methods within SCOUT, and thus, we account only for the number of times that consensus clustering is performed, since the actual computational complexity of performing each consensus clustering depends on the chosen clustering method.

**Exhaustive search.** This strategy aims to find a globally optimal solution under the chosen consensus clustering method by exhaustively searching through the space of all possible $T$s. There are $(k-1\choose l-1)$ ways to group all $k$ snapshots of $D$ into $l \in [1,k]$ segments. Thus, for all $l$s, the exhaustive search needs to explore the total of 
$$
\sum_{i=1}^{k} (k-1\choose i-1) = 2^{k-1}
$$
different segmentations (or, equivalently, change point sets).

To reduce the computational complexity, we use dynamic programming, as follows. The search contains $k$ iterations. Consider the $i^{th}$ iteration ($i \in [1,k]$). Let $D_{[q,r]} = \{G_q, G_{q+1}, \ldots, G_{r-1}, G_r\}$ be all consecutive snapshots of $D$ from time $q$ to time $r$, inclusively. The goal of the $i^{th}$ iteration is to solve the SCD problem for $D_{[0,i-1]}$ (i.e., for the first $i$ snapshots of $D$). For $i = k$, this means obtaining the solution for the whole network $D = D_{[0,k-1]}$. Recall from Appendix Section E.2.1...
that in order to solve the SCD problem, we first need to solve the CSCD problem for each possible number of segments (Appendix Equation E.10). That is, in the $i^{th}$ iteration, $\forall l \in [1, i]$, we need to find the optimal solution $O_{i,l}^*$ for $D_{[0,i-1]}$ that has $l$ segments. Next, we discuss how to find such $O_{i,l}^*$. Any solution $O$ for $D_{[0,i-1]}$ that has $l$ segments can be split into two parts with respect to start time $t$ of its last segment: 1) the part with the first $l-1$ segments, which can be thought of as a solution for prefix $D_{[0,t-1]}$ of $D_{[0,i-1]}$ and 2) the part with the last $l^{th}$ segment, which can be thought of as a solution for suffix $D_{[t,i-1]}$ of $D_{[0,i-1]}$. Conversely, solution $O$ can be constructed by combining the above two parts. Now, since $O_{i,l}^*$ is the optimal solution (with respect to the given consensus clustering method), its corresponding two parts should be optimal too. That is, the first part should be the optimal solution with $l-1$ segments for $D_{[0,t-1]}$ (which is exactly $O_{t,l-1}^*$, and which is known from the earlier $t^{(th)}$ iteration), and the second part should be the optimal solution with one segment for $D_{[t,i-1]}$ (which is a solution with just one segment, whose only segment partition can be obtained by performing consensus clustering of $D_{[t,i-1]}$). So, we know the first part of $O_{i,l}^*$ from one of the previous iterations and can compute its second part in the current iteration. However, in order to actually construct $O_{i,l}^*$ from the above two parts, we need to know start time $t$ of its last segment. If $l = 1$, there is only one value for $t$ (namely, $t = 0$), since the $l^{th}$ (i.e., the only) segment should encompass the whole $D_{[0,i-1]}$. If $l \in [2, i]$, $t$ can take any value from $l-1$ (in which case the $l-1$ segments in the first part are all singletons) to $i-1$ (in which case the $l^{th}$ segment in the second part is a singleton). So, for $l \in [2, i]$, to find $t$, we simply test all of its possible values and pick the one that produces the solution that maximizes the objective function (Appendix Figure E.1).

Given the above procedure, we next discuss its computational complexity. Consider the $i^{th}$ iteration ($i \in [1, k]$). In this iteration, we need to find $O_{i,l}^*$ for each $l \in [1, i]$. Recall from the above paragraph that each $O_{i,l}^*$ is constructed from two
parts, and only for the second part, corresponding to its last segment and starting
at some time point \( t \), we do not know the corresponding segment partition and thus
need to obtain this partition in the current iteration. For \( l = 1 \), we test only one
value of \( t \) (\( t = 0 \)), and for \( l \in [2, i] \), we test all values of \( t \) from \( l - 1 \) to \( i - 1 \). For
each \( t \), we need to obtain consensus partition for suffix \( D_{[t,i-1]} \) of \( D_{[0,i-1]} \). Clearly,
different values of \( l \) can deal with the same \( t \), i.e., the same suffix, and for a given
suffix we need to obtain its consensus partition only once. Overall, there are \( i \) dis-
tinct values of \( t \), from 0 to \( i - 1 \), and thus \( i \) distinct suffixes of \( D_{[0,i-1]} \) for which we
need to obtain consensus partitions. Thus, in the \( i^{th} \) iteration, we need to perform
consensus clustering for the total of \( i \) times, once for each such suffix. Therefore, for
all \( k \) iterations, we perform consensus clustering \( \sum_{i=1}^{k} i = \frac{k^2 + k}{2} = O(k^2) \) times.

Top-down search. This strategy aims to find a good solution by greedily searching
through the set of possible \( T \)s in a top-down manner. The search contains \( k \) iterations.
We start with one segment of length \( k \) and at each subsequent iteration split one of the
existing segments into two parts in a locally optimal way with respect to the chosen
objective function, until we reach \( k \) singleton segments (Appendix Figure E.2). Since
the search starts with one segment and since in each iteration the number of segments
is increased by one, the solution obtained in the \( i^{th} \) iteration is the solution for the
CSCD problem with \( i \) segments. Hence, after \( k \) iterations, we have one solution
for the CSCD problem for each possible number of segments, which can be used to
solve the SCD problem (Appendix Section E.2.1). More specifically, the top-down
search works as follows. In the first iteration, we have only one segment, and we
perform consensus clustering once for this segment. Then, at the start of the \( i^{th} \)
iteration (\( i \in [2, k] \)), we have \( i - 1 \) segments from the previous iteration (denoted as
\( \{s^{(i-1)}_0, s^{(i-1)}_1, \ldots, s^{(i-1)}_{i-2}\} \)), and we aim to split one of these \( i - 1 \) segments into two
parts by inserting a new change point \( t^*_{(i)} \), in order to produce \( i \) segments. There are
\( k - i + 1 \) candidate time points \( t' \) for \( t^*_{(i)} \); the total of \( k \) time points minus \( t_0 = 0 \) and
minus $i - 2$ change points selected in the previous iterations. Out of these candidates, we choose the one that maximizes gain (or minimizes loss) in our objective function.

To reduce the computational complexity, we show that we can reuse in each iteration the results from the previous iterations. In the $i^{th}$ iteration ($i \in [2, k]$), each candidate time point $t'$ leads to splitting some current segment $s_j^{(i-1)}$ into two parts. So, for a given $t$, we need to know two segment partitions: one for the first half of segment $s_j^{(i-1)}$ (that ends at time $t - 1$) and one for the second half of segment $s_j^{(i-1)}$ (that starts at time $t$). We do not necessarily need to compute these segment partitions in the current iteration, since we can reuse the results from the previous iterations, as follows. Consider the next $(i + 1)^{st}$ iteration. Let $s_{ri}^{(i)}$ be the segment that was split in the $i^{th}$ iteration (by inserting $t_{(i)}^*$). In the $(i + 1)^{st}$ iteration, all segments except the two resulted from splitting $s_{ri}^{(i-1)}$ are the same as in the $i^{th}$ iteration (Appendix Figure E.2). Thus, when testing candidate time points inside these unchanged segments, we can just reuse segment partitions from the previous iterations. So, in the $(i + 1)^{st}$ iteration, we only need to perform consensus clusterings for those candidate time points that are within the two newly created segments (Appendix Figure E.2). Such time points are all time points inside the segment that was split in the $i^{th}$ iteration except its start point and except already taken $t_{(i)}^*$.

The above described reuse of previous consensus clusterings generally allows for reducing the complexity compared to the exhaustive search. However, in the worst case, the complexity of the top-down search is still $O(k^2)$: if each new change point $t_{(i)}^*$ is selected as the earliest one (i.e., if $t_{(i)}^* = i - 1$), the overall number of performed consensus clusterings is $1 + 2(k - 1) + \sum_{i=3}^{k} (k - i + 1) = \sum_{i=1}^{k} (k - i + 1) = (k^2 + k)/2 = O(k^2)$. Nevertheless, even though this is the same theoretic complexity as for the exhaustive search above, in practice, the top-down search is faster (Appendix Section E.4).

**Bottom-up search.** This strategy aims to find a good solution by greedily searching
through the set of possible $T$s in a bottom-up manner. The search contains $k$ iterations. We start with $k$ singleton segments and at each subsequent iteration merge two existing adjacent segments in a locally optimal way (with respect to the chosen objective function), until we reach one large segment of length $k$. Since the search starts with $k$ segments and since in each iteration the number of segments is decreased by one, the solution obtained in the $i^{th}$ iteration is the solution for the CSCD problem with $k - i + 1$ segments. Hence, after $k$ iterations, we have one solution for the CSCD problem for each possible number of segments, which can be used to solve the SCD problem (Appendix Section E.2.1). More specifically, the bottom-up search works as follows. In the first iteration, we perform consensus clustering $k$ times, once for each segment. Then, at the start of the $i^{th}$ iteration ($i \in [2,k]$), we have $k - i + 2$ segments from the previous iteration (denoted as $\{s_{0}^{(i-1)}, s_{1}^{(i-1)}, \ldots, s_{k-i+1}^{(i-1)}\}$), and we aim to merge some two adjacent segments $(s_{r_{i}}^{(i)}, s_{r_{i}+1}^{(i)})$ from these $k - i + 2$ segments in order to produce $k - i + 1$ segments. There are $k - i + 1$ candidate segment pairs $(s_{j}^{(i)}, s_{j+1}^{(i)})$ (or, equivalently, $k - i + 1$ candidate change points to be removed), since there are $k - i + 2$ segments and since we consider only adjacent segment pairs. Out of these candidates, we choose the one that maximizes gain (or minimizes loss) in our objective function.

To reduce the computational complexity, we next show that we can reuse in each iteration the results from the previous iterations. In the $i^{th}$ iteration ($i \in [2,k]$), each candidate segment pair $(s_{j}^{(i)}, s_{j+1}^{(i)})$ leads to merging segments $s_{j}^{(i)}$ and $s_{j+1}^{(i)}$. So, for a given segment pair $(s_{j}^{(i)}, s_{j+1}^{(i)})$, we need to know one segment partition for the merged segment $s_{q}^{(i)} \cup s_{q+1}^{(i)}$. We do not necessarily need to compute these segment partitions in the current iteration, since we can reuse the results from the previous iterations, as follows. Consider the next $(i + 1)^{st}$ iteration. In the $(i + 1)^{st}$ iteration, all segments except the newly created one (i.e., the segment resulting from merging the two segments chosen in the $i^{th}$ iteration) are the same as in the $i^{th}$ iteration.
Thus, when testing candidate segment pairs not involving the new segment, we can just reuse segment partitions from the previous iteration. So, in the \((i + 1)^{st}\) iteration, we only need to perform consensus clusterings for those candidate segment pairs that involve the new segment. There are at most two such segment pairs, since we consider only adjacent segments.

In the first iteration, we perform consensus clustering \(k\) times (once for each snapshot). In the second iteration, we perform consensus clustering \(k - 1\) times (once for each pair of adjacent snapshots). For all subsequent iterations, as discussed above, we perform consensus clustering at most twice. Therefore, for all \(k\) iterations, we need to perform consensus clustering at most \(k + (k - 1) + \sum_{i=3}^{k} 2 = 4k - 5 = O(k)\) times.

E.3 Experimental Setup

E.3.1 Methods for Comparison

Here, we expand our discussion from Section 6.2.4.1 and discuss the methods that we use in our experiments and their parameters.

**GraphScope** does not accept any user-defined parameters. Note that GraphScope was originally designed to work only with bipartite graphs, and thus it produces two separate partitions. Hence, in order to handle unipartite graphs such as the data from our study, we constrain GraphScope to produce only one partition \[199\]. The method can solve only the SCD problem. We use a publicly available implementation of GraphScope \[18\].

**Multi-Step** performs an agglomerative clustering of the snapshots, merging them into segments to produce a hierarchical tree. To get a solution for the CSCD problem, we cut the tree at the level that results in the desired number of segments. To get a solution for the SCD problem, we first test Multi-Step’s suggested procedure of
cutting the tree at the level above which the highest segment similarity is negative. We note that this procedure is used as a way to stop the merging process early in order to prevent meaningless merges, rather than as a way to select the best segmentation \cite{11}. Importantly, as we show in Section 6.3 this procedure consistently underestimates the number of segments that actually exist in the data. To address this, we introduce a user-specified segment similarity threshold $\theta$, and instead of stopping the merging process as soon as the highest similarity becomes negative, we instead stop this process as soon as the highest similarity becomes less than $\theta$. That is, in Multi-Step’s default procedure, $\theta = 0$. To give Multi-Step the best-case advantage, we vary $\theta$ from 0 to 2 in increments of 0.1 (note that since the similarity between two segments is computed as the sum of modularities of the two segments, and since the maximum value of modularity for any segment is 1, the maximum possible segment similarity value is 2; Appendix Section E.1). However, we find that the optimal threshold $\theta$ is network-specific, and a threshold that works well for one network may not work well for other networks (Section 6.2.4.1). Note that by trying different values of $\theta$, essentially, we allow Multi-Step to try solutions with different numbers of segments. Thus, when the ground truth number of segments is known (see below), we simply provide this information as input to Multi-Step. That is, instead of using Multi-Step to solve the SCD problem, we use it to solve the CSCD problem where we set $l$ to match the ground truth number of segments. This is not fair to the other methods (including SCOUT), which aim to solve the full SCD problem (and thus automatically find $l$ that ideally matches the ground truth value). Yet, this is what we have to do in order to include Multi-Step into the comparison, since this method can reliably solve only the CSCD problem. We use a publicly available implementation of Multi-Step \cite{11}. 

**GHRG** relies on a sliding window approach with the length of the window $w$ being a user-defined parameter. We test $w \in \{4, 8, 12\}$. For each segment, GHRG
results in a generalized hierarchical tree model instead of a partition. To obtain a partition from this model, we cut the tree in a way that maximizes modularity \([22]\). The method can solve only the SCD problem. We use a publicly available implementation of GHRG \([156]\).

**SCOUT** contains three main components: objective function, consensus clustering, and search strategy. We test different choices for these components: two objective functions (with four choices for \(Q_P\) and two choices for \(Q_B\); Appendix Section \[E.2.1\]), three consensus clustering approaches (with seven choices for each of sum graph and consensus matrix and one choice for Average-Louvain; Appendix Section \[E.2.2\]), and three search strategies (Appendix Section \[E.2.3\]). To allow for experimenting with the different parameter choices, our initial SCOUT implementation focuses on flexibility (to allow for easily testing various parameter choices for the method’s components) rather than on running time. However, once we finalize the most optimal (i.e., accurate yet efficient) choice of the parameters (Appendix Section \[E.4\]), we develop a faster parallel SCOUT implementation tailored for the selected parameters and aimed at reducing the running time. So, when comparing SCOUT against the other approaches, we use its latter fast implementation. As discussed in Appendix Section \[E.2.1\], SCOUT can solve both the CSCD and SCD problem.

### E.3.2 Datasets

Here, we expand our discussion from Section \[6.2.4.2\] and describe in detail the four steps of synthetic network generation (Appendix Figure \[E.4\]). Recall that our model has the following parameters: the number of snapshots \(k\), the number of segments \(l\), the number of nodes in each snapshot \(n\), the minimum possible number of nodes in each cluster \(c_{\text{min}}\), and the two parameters \(c_{\text{in}}\) and \(c_{\text{out}}\) controlling intra- and inter-community edge density of the snapshots. As described in Section \[6.2.4.2\] the goals
of the steps are: 1) create $T^{(gt)}$, 2) create a special auxiliary graph $G_P$, 3) use $G_P$ to create $P^{(gt)}$, and 4) use $T^{(gt)}$ and $P^{(gt)}$ to generate $D$. The details of the four steps are as follows:

1) To create a segmentation with $l$ segments, we randomly sample $l-1$ change points $T^{(gt)}$ from $[1,k-1]$. For example, in Appendix Figure E.4 we select two change points $t_1^{(gt)}$ and $t_2^{(gt)}$ to create three segments. Thus, at the end of this step, we have the change point set $T^{(gt)}$.

2) In this and the next step, we aim to generate segment partitions $P^{(gt)}$, as follows. To achieve this, we use a special partition graph $G_P$. In this step, we intuitively define $G_P$ and describe how we generate $G_P$ and in the next step we describe how use $G_P$ to create $P^{(gt)}$.

We need to create $G_P = (V_P, E_P)$ as a weighted directed $l$-partite graph (i.e., $V_P = V_P^{(0)} \cup V_P^{(1)} \cup \cdots \cup V_P^{(l-1)}$) with every edge in $E_P$ having form $(u,v)$, where $u \in V_P^{(i-1)}$ and $v \in V_P^{(i)}$ for some $i \in [1,l-1]$. To avoid confusion between nodes of $G_P$ and nodes of $D$, we refer to nodes of $G_P$ as supernodes. Intuitively, once we generate $G_P$ at the end of this step, each of its supernode sets $V_P^{(i)}$ will correspond to segment partition $p_i^{(gt)}$ of $O^{(gt)}$, with each supernode of $V_P^{(i)}$ corresponding to some cluster of $p_i^{(gt)}$. For example, in Appendix Figure E.4 $V_P^{(0)}$ has four supernodes, so $p_0^{(gt)}$ will have four clusters. Each edge between two supernodes of $G_P$ intuitively means that the two clusters corresponding to these two supernodes have shared members (see the next step for more details).

We construct $G_P$ starting with empty sets $V_P$ and $E_P$. To create $V_P$, we add a random number $r_i \in [2, \lfloor n/c_{min} \rfloor]$ of supernodes to each $V_P^{(i)} \subset V_P$, $i \in [0,l-1]$. Recall from above that $c_{min}$ is the user-defined minimum required number of nodes in a cluster. Thus, the limits for $r_i$ mean that we want segment partition corresponding to $V_P^{(i)}$ to have at least two clusters, while at the same time we want to prevent it from having too many clusters. To create $E_P$, we randomly create edges between
each pair of adjacent sets $V_{P}^{(i-1)}$ and $V_{P}^{(i)} \ (i \in [1, l-1])$ as follows. For each edge, we randomly pick its two endpoints: one supernode in $V_{P}^{(i-1)}$ and one supernode in $V_{P}^{(i)}$. When adding edges to $E_{P}$, we aim to satisfy the following three conditions: 

a) the edges do not form a perfect matching between $V_{P}^{(i-1)}$ and $V_{P}^{(i)}$ (i.e., segment partitions $p_{i-1}^{(gt)}$ and $p_{i}^{(gt)}$ are not identical), b) each supernode in $V_{P}^{(i-1)}$ has at least one outgoing edge and each supernode in $V_{P}^{(i)}$ has at least one incoming edge (i.e., clusters cannot appear/disappear), and c) for each edge $e = (u, v)$, assuming $d^{+}$ and $d^{-}$ is the outdegree and indegree of a supernode, respectively, one of the following holds: (i) $d^{+}(u) = 1$ and $d^{-}(v) > 1$ (i.e., the cluster corresponding to $u$ merges with at least one other cluster into the cluster corresponding to $v$; e.g., the top right edge in Appendix Figure E.4), (ii) $d^{+}(u) > 1$ and $d^{-}(v) = 1$ (i.e., the cluster corresponding to $u$ is split into several clusters including the cluster corresponding to $v$; e.g., the bottom right edge in Appendix Figure E.4), or (iii) $d^{+}(u) = d^{-}(v) = 1$ (i.e., the cluster corresponding to $u$ and the cluster corresponding to $v$ have the same members; e.g., the top left edge in Appendix Figure E.4). Intuitively, for two adjacent segment partitions, edges satisfying (i) correspond to merges of clusters, edges satisfying (ii) correspond to splits of cluster, and edges satisfying (iii) correspond to unchanged clusters. Note that in general, the condition (iii) can be relaxed; we include it to provide further coherence between individual clusters so that their changes can be described via three simple events (merge, split, and continuation).

3) Using $G_{P}$ from the previous step, we now generate the set of segment partitions $P^{(gt)}$, as shown in Appendix Algorithm E.1. Intuitively, each edge in $G_{P}$ means that some two clusters share members. We construct segment partitions one by one, starting from the partition of the first segment. For each segment partition, we determine the membership of a given cluster based on the incoming edges of the corresponding supernode and the previous segments partition. At the end of this step, we have the complete $O^{(gt)} = (T^{(gt)}, P^{(gt)})$. 

261
Algorithm E.1 Step 3 of our synthetic network generation process

**Input:** $G_P$  
**Output:** $P^{(gt)}$

1: $V \leftarrow [0, n - 1]$  
2: $p_0^{(gt)} \leftarrow RandomPartition(V, |V_P^{(0)}|)$  
3: for $i \in [1, l - 1]$ do  
   4: for $v \in V^{(i)}_P$ do  
      5: $c(v) \leftarrow \emptyset$  
   6: end for  
   7: for $u \in V^{(i-1)}_P$ do  
      8: $p^{(temp)} \leftarrow RandomPartition(c(u), d^+(u))$  
      9: $j \leftarrow 0$  
     10: for $c^{(temp)} \in p^{(temp)}$ do  
         11: $c(Outneighbors(u)[j]) \leftarrow c(Outneighbors(u)[j]) \cup p^{(temp)}$  
         12: $j \leftarrow j + 1$  
     13: end for  
   14: end for  
15: $p_i^{(gt)} \leftarrow \{c(v) | v \in V^{(i)}_P\}$  
16: end for  
17: $P^{(gt)} \leftarrow \{p_i^{(gt)} | i \in [0, l - 1]\}$  
18: return $P^{(gt)}$

4) We use $O^{(gt)} = (T^{(gt)}, P^{(gt)})$ to generate snapshots of $D$ as follows. Intuitively, we aim to generate each snapshot based solely on the corresponding segment partition. Thus, to generate a snapshot $G_j$ of a given segment $s_i$, we use the stochastic blockmodel with partition $p_i^{(gt)}$ and stochastic block matrix $\theta$ (Appendix Section E.2.1). We use the same $\theta$ for all segments, with $\theta_{cucv} = c_{in}/n$ if $c_u \neq c_v$ and $\theta_{cucv} = c_{in}/n$ otherwise. So, to generate a snapshot $G_j$ of segment $s_i$, for each pair of nodes in $G_j$, we independently place an edge between the two nodes with probability $c_{in}/n$ if they are in the same cluster in $p_i$ and with probability $c_{out}/n$ otherwise \[146\]. Note that even though all snapshots in segment $s_i$ are created based on the same segment partition $p_i$, they still likely differ from each other due to randomness in the stochastic blockmodel. At the end of this step, we have all snapshots $G_j$ of $D$, and, moreover, the structure of $D$ reflects the ground truth $O^{(gt)} = (T^{(gt)}, P^{(gt)})$.  

262
E.3.3 Evaluation Measures

Here, we expand our discussion from Section 6.2.4.3. We start by discussing the three output similarity measures.

**Segmentation similarity Sim\(_T\).** Formally, for a given \(O = (T, P)\), we construct its time point partition \(\mathcal{P}_T\) as a partition of the set \([0, k - 1]\) into \(l\) clusters \(c_i, i \in [0, l - 1]\), where cluster \(c_i = \{t_i, t_i + 1, \ldots, t_{i+1} - 1\}\). Then, to compute Sim\(_T\) between \(O^*\) and \(O^{(gt)}\), we measure similarity of their corresponding time point partitions \(\mathcal{P}_T^*\) and \(\mathcal{P}_T^{(gt)}\), as follows:

\[
Sim_T(O^*, O^{(gt)}) = H(\mathcal{P}_T^*, \mathcal{P}_T^{(gt)}),
\]

(E.11)

where \(H\) can be any partition similarity measure.

**Partition similarity Sim\(_P\).** Formally, to compute Sim\(_P\) between \(O^*\) and \(O^{(gt)}\), we introduce average snapshot partition similarity, as follows:

\[
Sim_P(O^*, O^{(gt)}) = \frac{1}{k} \sum_{j=0}^{k-1} H(p^*_{seg(j,T^*)}, p^{(gt)}_{seg(j,T^{(gt)})}),
\]

(E.12)

where \(H\) can be any partition similarity measure and \(seg(j, T)\) is the function that returns the index of the segment containing snapshot \(G_j\) under the segmentation induced by change point set \(T\) (i.e., \(seg(j, T) = i \iff G_j \in s_i\)). Note that even though Sim\(_P\) focuses on the partition aspect of the SCD problem, it still implicitly relies on the segmentation aspect via the above \(seg\) function.

**Overall similarity Sim\(_B\).** Formally, for a given \(O = (T, P)\), we construct a node-time partition \(\mathcal{P}_B\) as a partition of the set \(\{(u, t)|u \in V_t, t \in [0, k - 1]\}\) into \(\sum_{i=0}^{l-1} |p_i|\) clusters. Two node-time pairs \((u_1, t_1)\) and \((u_2, t_2)\) are clustered together in \(\mathcal{P}_B\) if their time points \(t_1\) and \(t_2\) belong to the same segment (i.e., if \(t_1\) and \(t_2\) are in the same cluster in \(\mathcal{P}_T\), or \(seg(t_1, T) = seg(t_2, T)\)) and if their nodes \(u_1\) and \(u_2\) belong to
the same cluster in the corresponding segment partition (i.e., if $u_1$ and $u_2$ are in the same cluster in $p_{\text{seg}(t_1,T)}$). Then, to compute $\text{Sim}_B$ between $O^*$ and $O^{(\text{gt})}$, we measure similarity between their corresponding node-time partitions $\mathcal{P}_B^*$ and $\mathcal{P}_B^{(\text{gt})}$, as follows:

$$\text{Sim}_B(O^*, O^{(\text{gt})}) = H(\mathcal{P}_B^*, \mathcal{P}_B^{(\text{gt})}), \quad \text{(E.13)}$$

where $H$ can be any partition similarity measure.

Next, we describe the four partition similarity measures $H$ that we use: 1) Normalized Mutual Information (NMI) [212] – a measure of similarity based on the mutual information (MI), normalized to have values in $[0,1]$. 2) Adjusted Mutual Information (AMI) [212] – an adjusted for chance version of MI. 3) Adjusted Rand Index (ARI) [212] – an adjusted for chance version of the Rand Index, a measure of similarity based on counting pairs of observations assigned to the same cluster or different clusters in two partitions. For AMI and ARI, the adjustment for chance means correction for chance agreement between two partitions [212]. 4) V-Measure (VM) [175] – the harmonic mean of homogeneity (whether the first partition groups together only those objects that are grouped together in the second partition) and completeness (whether the first partition groups together all those objects that are grouped together in the second partition). Note that for all of the above measures, a higher value means higher similarity, with two identical partitions having similarity of one.

II) Next, we describe the three classification accuracy measures that we use: 1) $\forall i \in [1,k]$, precision is the fraction of the top ranked $i$ time points that are ground truth change points (i.e., that belong to $T^{(\text{gt})}$), and recall is the fraction of all ground truth change points that are among the top ranked $i$ time points. To summarize the values of precision and recall over all possible values of $i$, we compute the area under the precision-recall curve (AUPR). 2) $\forall i \in [1,k]$, F-score is the harmonic mean of precision and recall (that is, F-score balances the two quantities). We report the
maximum F-score over all values of $i$. 3) $\forall i \in [1,k]$, sensitivity is equal to recall and specificity is the fraction of ground truth non-change points (corresponding to the complement of $T^{(gt)}$) that are not among the top ranked $i$ time points. To summarize the values of sensitivity and specificity over all possible values of $i$, we compute the area under the receiver operator characteristic curve (AUROC).

We obtain the ranked list of all time points for each of the considered methods as follows. For a given method, to get its ranked list, we compute the score (see below) for each time point $t \in [1,k-1]$, such that the time points with lower scores are ranked higher (i.e., are more “change point-like”). Note that here we exclude from consideration time point $t_0 = 0$, because, by definition, for any method, $t_0$ always denotes the start of the first segment, and hence including it into comparison does not provide any method-specific information. If a method is capable of solving the CSCD problem, we compute the scores of the time points using the solutions with all possible numbers of segments for the CSCD problem. Specifically, let $O^*_i = (T^*_i, P^*_i)$ be the solution with $i$ segments for the CSCD problem. Intuitively, if a given time point $t$ is a change point in $O^*_i$ (i.e., if $t \in T^*_i$), this means that the method, when asked to select only $i - 1$ change points (i.e., to produce $i$ segments), chooses $t$ as one of these $i - 1$ change points. Hence, to capture the intuition that more “change point-like” time points appear in the solutions with smaller number of segments, we compute the score of a given time point $t$ as the smallest number of segments for which $t$ appears as a change point in the corresponding CSCD solution (i.e., $score(t) = \min\{i| i \in [1,k], t \in T^*_i\}$). So, the highest ranked time point will appear in the solution with two segments (for which only one time point is selected as change point), while the lowest ranked time point will appear only in the solution with $k$ segments (for which all time points are selected as change point). Since among the considered methods only Multi-Step and SCOUT can solve the CSCD problem (Appendix Section E.3.1), the above procedure can only be used for these.
two methods. For the remaining two methods, GraphScope and GHRG, we use alternative strategies for extracting their ranked lists, as follows. GraphScope, at each step, marks a time point \( t \) as a change point if the community organization of \( G_t \) does not match well the community organization of the current segment (Appendix Section E.1). The match is measured via the MDL principle, as the difference of the cost of encoding the current segment and \( G_t \) together and the cost of encoding them separately. Intuitively, the smaller the difference, the “cheaper” it is to add \( G_t \) to the current segment, and when the difference is negative, \( G_t \) is added to the current segment. Thus, we use the difference of the encoding costs as the score of \( t \). GHRG performs a statistical test at each step to determine whether the current window contains a change point (Appendix Section E.1). A change point is detected if its corresponding \( p \)-value is smaller than the chosen threshold. Intuitively, the smaller the \( p \)-value, the more confident the method is that the given time point \( t \) is a change point. Thus, we use the \( p \)-value as the score of \( t \).

E.3.4 Statistical Significance of Two Methods’ Performance Difference

Given a synthetic network configuration, evaluation measure, and a pair of methods, we compute the statistical significance of the difference between the performance of the two methods as follows. For each method, we create a list containing performance scores of the method for all network instances of the synthetic network configuration. Since we know which score in a given list corresponds to which network instance, we use paired \( t \)-test to compute the statistical significance of the difference between the two lists produced by the two methods. There are four possible outcomes: 1) method 1 outperforms method 2, and the improvement is statistically significant, 2) method 1 outperforms method 2, but the improvement is not statistically significant, 3) method 2 outperforms method 1, but the improvement is not statistically significant, and 4) method 2 outperforms method 1, and the improve-
ment is statistically significant (note that outcomes 2 and 3 also cover the case when the two methods’ scores are tied). By statistically significant, we mean that the p-value that results from paired t-test is below a threshold. We test three p-value thresholds: 0.05, 0.01, and 0.001.

E.4 Results: the Effect of Method Parameter Choices

Here, we expand our discussion from Section 6.3.1 and discuss the effect of SCOUT parameters. We test the effect on the method’s performance of \( a) \) the objective function, \( b) \) consensus clustering method, and \( c) \) search strategy.

\( a) \) Objective function is used twice (Appendix Section E.2.1): 1) when computing the best solution for each possible number of segments (Appendix Equation E.9), and 2) when choosing among these best solutions the final one (Appendix Equation E.10). The objective functions can differ between the two cases: in case “1”, we can use any \( Q_P \) or any \( Q_B \), and in case “2”, we have to use a \( Q_B \) (Appendix Section E.2.1). Thus, since SCOUT uses \( Q_B \) up to two times, while it uses \( Q_P \) up to one time, we first test the effect of \( Q_B \). Recall that we evaluate two \( Q_B \) measures: one based on BIC and the other based on AIC (Appendix Section E.2.1). In general, BIC results in higher \( Sim_B \) compared to AIC (Appendix Figure E.7(a)). The reason for this is that AIC produces more segments than BIC, usually overestimating the ground truth number of segments (Appendix Figure E.7(b)). Recall that this behavior of AIC is not surprising (Appendix Section E.2.1). So, we focus on \( Q_B \) based on BIC. This gives us the choice of \( Q_B \) for case “2”. For case “1”, we can use \( Q_B \) based on BIC or one of the four \( Q_P \)s (based on modularity, conductance, normalized cut, or average-ODF; Appendix Section E.2.1). Hence, we next test the effect of \( Q_P \) versus \( Q_B \) in case “1”. Out of all \( Q_P \)s, modularity generally leads to the highest \( Sim_B \) (Appendix Figure E.8). However, the best results in terms of \( Sim_B \) are achieved when using \( Q_B \) based on BIC and not \( Q_P \) based on modularity (Appendix Figure E.8). So, whether
we are considering case “1” or case “2”, $Q_B$ based on BIC overall outperforms all other tested objective functions. Thus, we focus on $Q_B$ based on BIC as SCOUT’s objective function.

b) Consensus clustering is used to produce segment partitions, given a segmentation. We use three general types of consensus clustering approaches: sum graph, Average-Louvain, and consensus matrix (Appendix Section E.2.2). Recall that the sum graph and consensus matrix approaches are parameterized with the static clustering method (Appendix Section E.2.2). So, before we compare the above three general types of approaches, we first aim to choose the best static clustering method for sum graph and consensus matrix approaches. We evaluate seven static clustering methods: Fast Modularity, Label Propagation, Leading Eigenvector, Infomap, Walktrap, Louvain, and Stabilized Louvain (Appendix Section E.2.2). We find that generally Walktrap works the best in terms of $Sim_B$ while having comparable running time (Appendix Figure E.9). So, we focus on Walktrap as the static clustering method for both sum graph and consensus matrix approaches. Next, we compare the three general approach types. In terms of $Sim_B$, all three approaches lead to comparable results (Appendix Figure E.10(a)), even though Average-Louvain and consensus matrix are more sophisticated compared to sum graph and thus would be expected to be superior. In terms of the running time, sum graph is the fastest of the three approaches (Appendix Figure E.10(b)). Thus, we focus on sum graph with Walktrap as SCOUT’s consensus clustering approach.

c) Search strategy is used to determine how SCOUT searches through the space of possible segmentations. We evaluate three strategies: the exhaustive search, top-down search, and bottom-up search (Appendix Section E.2.3). In terms of $Sim_B$, all three strategies lead to comparable results (Appendix Figure E.11(a)). In terms of the running time, bottom-up search is the fastest one (Appendix Figure E.11(b)). Thus, we focus on the bottom-up search as SCOUT’s search strategy.
Figure E.1: The procedure of obtaining solution $O^*_{i,t}$ for $D_{[0,i-1]}$ that has $l$ segments during the $i^{th}$ iteration of the exhaustive search. For this, we construct a set of candidate solutions (shown in rows) and pick as $O^*_{i,t}$ the one solution from this set that maximizes the objective function. Each candidate solution is obtained by combining: 1) solution $O^*_{i-1,t-1}$ that has $l-1$ segments, which was obtained in one of the previous iterations (shown in blue), and 2) a solution with one segment, whose only segment partition is obtained during the current iteration (shown in orange).
Figure E.2: The top-down search. The procedure starts with one segment encompassing the whole network and then iteratively splits one of the current segments into two. For the $i^{th}$ iteration, the orange box indicates the segment that was split as a result of this iteration, and blue boxes indicate all other, unchanged segments.

Figure E.3: The bottom-up search. The procedure starts with $k$ singleton segments and then iteratively merges two of the current adjacent segments into one. For the $i^{th}$ iteration, the orange box indicates the segment that was created as a result of this iteration, and blue boxes indicate all other, unchanged segments.
Figure E.4: The process of constructing synthetic dynamic network $D$ with known ground truth solution $O^{(gt)} = (T^{(gt)}, P^{(gt)})$. The four steps of the process, described in the text, are illustrated from top to bottom.
Figure E.5: The effect of $\theta$ value for Multi-Step, in terms of (a) $Sim_B$ and (b) the number of segments $l$, as we vary the ground truth synthetic network configuration ($x$-axis). Note that we run the analysis for $\theta \in [0, 2]$ in increments of 0.1. However, since the results for $\theta \geq 1.0$ are all the same, we show the results only for $\theta \leq 1.0$, and for visual clarity, we show the results in increments of 0.2. In panel (b), for a given ground truth configuration, the dotted line corresponds to the ground truth number of segments.

Figure E.6: The effect of $w$ value for GHRG, in terms of $Sim_B$, as we vary the ground truth synthetic network configuration ($x$-axis).
Figure E.7: The effect of $Q_B$ choice for SCOUT when choosing the optimal number of segments, in terms of (a) $\text{Sim}_B$ and (b) the number of segments $l$, as we vary the ground truth synthetic network configuration (x-axis). Here, in all cases, we fix consensus clustering method as sum graph with Walktrap and search strategy as exhaustive search. In panel (b) for a given ground truth configuration, the dotted line corresponds to the ground truth number of segments.
Figure E.8: The effect of $Q$ choice for SCOUT when producing the best solution for each possible number of segments, in terms of $Sim_B$, as we vary the ground truth synthetic network configuration ($x$-axis). Note that when using $Q_P$ (the first four series), we still need to choose the optimal number of segments, which we do using $Q_B$ (i.e., BIC). Here, in all cases, we fix consensus clustering method as sum graph with Walktrap, and we fix search strategy as exhaustive search.
Figure E.9: The effect of the choice of static clustering method in the sum graph consensus clustering method for SCOUT, in terms of (a) $Sim_B$ and (b) running time, as we vary the ground truth synthetic network configuration ($x$-axis). Here, in all cases, we fix search strategy as exhaustive search.
Figure E.10: The effect of the choice of consensus clustering method for SCOUT, in terms of (a) $Sim_B$ and (b) running time, as we vary the ground truth synthetic network configuration (x-axis). Here, in all cases, we fix search strategy as exhaustive search.

Figure E.11: The effect of the choice of search strategy for SCOUT, in terms of (a) $Sim_B$ and (b) running time (logarithmic scale), as we vary the ground truth synthetic network configuration (x-axis).
Figure E.12: Rankings of the methods with respect to Sim\(T\) and change point classification. Since GHRG could not be run for the larger networks, the results are split into those for the configurations with 50 and 100 nodes per snapshot (top) and those for the configurations with 500 and 1000 nodes per snapshot (bottom). Note that for change point classification, we exclude from consideration configurations with the minimum (i.e., one) and maximum (i.e., 16) possible numbers of ground truth segments. This is because for these configurations, either there are no change points at all (i.e., for one segment) or every time point is a change point (i.e., for 16 segments), so change point classification cannot be performed. The rankings are computed as follows. For each synthetic network configuration, we compare the four methods’ scores (average scores over all instances of the given configuration) to identify the first, second, third, and fourth best method; ties are allowed, in which case, two methods would be assigned the same rank. Then, we summarize these results over all considered synthetic network configurations by measuring, for each method (x-axis), how many times the given method is ranked as the first, second, third, and fourth best method (expressed as the percentage of all considered configurations; y-axis). “N/A” indicates that either a given method could not be run (which is the case for GHRG for the larger networks) or it was excluded from the consideration (which is the case for Multi-Step in panel (a)). The reason we exclude Multi-Step from the consideration in certain configurations in panel (a) (namely, those with the minimum and maximum possible numbers of segments) is as follows. We provide Multi-Step with the ground truth number of segments as input, and there is only one possible segmentation with minimum or maximum possible number of segments, so Multi-Step trivially returns the perfect Sim\(T\) score for these extreme configurations.
Figure E.13: SimT and SimP scores for all 20 synthetic network configurations. The rows correspond to four different numbers of nodes per snapshot, with the number of nodes increasing from top to bottom. The columns correspond to five different numbers of ground truth segments: (a) one ground truth segment, (b) two ground truth segments, (c) four ground truth segments, (d) eight ground truth segments, and (e) 16 ground truth segments. In each panel, for each method, the results are averaged over all of the corresponding synthetic network instances.
Figure E.14: The number of segments $l$ in the solutions produced by the methods for synthetic networks with (a) 50-node snapshots, (b) 100-node snapshots, (c) 500-node snapshots, and (d) 1000-node snapshots. In each panel, the results are grouped by the number of ground truth segments and averaged over all of the corresponding synthetic network instances. A given dotted line corresponds to the ground truth number of segments in the given synthetic network configuration. Note that for Multi-Step, we use the default parameters, since when using Multi-Step $\star$ that is provided with the number of ground truth segments as input, this method trivially returns the correct number of ground truth segments.
Figure E.15: Method comparison for synthetic networks with 50, 500, and 1000 nodes per snapshot (shown from top to bottom) with respect to (a) change point classification, (b) \(Q_P\), and (c) \(Sim_B\). For a given ground truth configuration, the results are averaged over all of the corresponding synthetic network instances. In panel (b), the dotted lines correspond to the ground truth score. Note that for panel (a) we exclude from consideration the configurations with the minimum and maximum possible numbers of ground truth segments. We do this because for these configurations, either there are no change points at all (for one segment) or every time point is a change point (for 16 segments), which means that change point classification cannot be performed. GHRG could not be run for two largest network sizes due to its high computational complexity. Equivalent results for synthetic networks with 100 nodes per snapshot are shown in Figure 6.5.
Figure E.16: Rankings of the methods with respect to (a) $Q_P$ and (b) $Sim_P$. Since GHRG could not be run for the larger networks, the results are split into those for the configurations with 50 and 100 nodes per snapshot (top) and those for the configurations with 500 and 1000 nodes per snapshot (bottom). The rankings are computed as follows. For each synthetic network configuration, we compare the four methods’ scores (average scores over all instances of the given configuration) to identify the first, second, third, and fourth best method; ties are allowed, in which case, two methods would be assigned the same rank. Then, we summarize these results over all considered synthetic network configurations by measuring, for each method (x-axis), how many times the given method is ranked as the first, second, third, and fourth best method (expressed as the percentage of all considered configurations; y-axis). “N/A” indicates that the given method could not be run (which is the case for GHRG for the larger networks).
Figure E.17: The number of segments $l$ in the solutions produced by the methods for real-world networks.
Figure E.18: $Q_P$ scores of 1) SCOUT’s solutions for different numbers of segments $l$ and 2) the solutions of the existing methods for (a) Hypertext network, (b) AMD Hope network, (c) Reality Mining network, and (d) Enron network. For SCOUT, the line shows its $Q_P$ score when solving the CSCD problem while varying the number of segments. For each of the existing methods, the mark shows $Q_P$ score of its solution, with the position of the mark along the $x$-axis corresponding to the number of segments $l$ in the solution. Equivalent results for the remaining real-world networks are shown in Figure 6.10.
TABLE E.1

REAL-WORLD NETWORKS THAT WE USE IN OUR STUDY

<table>
<thead>
<tr>
<th>Name</th>
<th># of nodes</th>
<th># of edges</th>
<th>Time span</th>
<th>Edge type</th>
<th># of nodes</th>
<th># of edges</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypertext</td>
<td>113</td>
<td>21K</td>
<td>3 days</td>
<td>Proximity</td>
<td>47 (20)</td>
<td>73 (53)</td>
<td>30 min</td>
</tr>
<tr>
<td>AMD Hope</td>
<td>409</td>
<td>1.26M</td>
<td>3 days</td>
<td>Co-location</td>
<td>161 (101)</td>
<td>4.5K (3.9K)</td>
<td>1 hour</td>
</tr>
<tr>
<td>High School</td>
<td>327</td>
<td>189K</td>
<td>5 days</td>
<td>Proximity</td>
<td>227 (39)</td>
<td>499 (237)</td>
<td>1 hour</td>
</tr>
<tr>
<td>Reality Mining</td>
<td>78</td>
<td>5K</td>
<td>10 months</td>
<td>Phone call</td>
<td>28 (11)</td>
<td>25 (12)</td>
<td>1 week</td>
</tr>
<tr>
<td>Enron</td>
<td>184</td>
<td>121K</td>
<td>2.5 years</td>
<td>Email</td>
<td>99 (40)</td>
<td>267 (154)</td>
<td>1 month</td>
</tr>
<tr>
<td>Senate</td>
<td>51</td>
<td>28K</td>
<td>227 years</td>
<td>Voting similarity</td>
<td>35 (15)</td>
<td>248 (268)</td>
<td>2 years</td>
</tr>
</tbody>
</table>

* “Network” columns show general information about a given network. “Snapshots” columns show properties of the network snapshots averaged over all snapshots.*
BIBLIOGRAPHY


